

PROGRAMACION PARA SUPERUSUARIOS
PROBLEMAS
PROBLEMAS
PROBLEMAS
AMSTRAD
SOLUCIONES
SOLUCIONES
SOLUCIONES
W. Johnson
ra-ma

PROGRAMACION PARA SUPERUSUARIOS

PROBLEMAS
PROBLEMAS
PROBLEMAS
AMSTRAD
SOLUCIONES
SOLUCIONES
SOLUCIONES

W. Johnson



PROGRAMACIÓN PARA SUPERUSUARIOS

Título de la obra original:

POWERFUL PROGRAMMING FOR AMSTRADS

Publicado en el Reino Unido por:
SIGMA PRESS

Traducción: Ramona Pozuelo Verdasco
Producción: Vector Ediciones S.A.

Editado por:

RA-MA Editorial
Ctra. de Canillas, 144
28043 Madrid - España
Telefs. 200 97 46/47

ISBN 84 86381 14 2
Depósito Legal: M. 3.895 - 1987
Imprime Signo Impresores, S. A.
Albasanz, 27. 28037 Madrid.

Prefacio

Este conjunto de subrutinas ha sido reunido para su publicación a partir de una serie de programas escritos en BASIC destinados a resolver problemas de ingeniería, manipulación de imagen y manejo y almacenamiento de datos. Debería resultar muy interesante para todos los usuarios de ordenadores, ya que contiene útiles rutinas de referencia, así como rutinas difíciles de encontrar o muy pesadas de escribir.

Las rutinas están escritas en Amstrad BASIC, pero se las puede traducir fácilmente a otros dialectos de BASIC; además, merece la pena estudiar detenidamente las técnicas empleadas en muchos de los programas. Produce una gran satisfacción conseguir una solución elegante a un problema, y el tema subyacente es que un programa debería ser un cuerpo de instrucciones con subrutinas eficaces. La eficacia hay que entenderla en términos de tiempo de ejecución, necesidades de memoria, no ocurrencia de errores y exactitud de los resultados, aunque la presentación de éstos últimos también es importante.

El buen programador es aquel al que no le satisface fácilmente un programa y siempre busca formas de mejorarlo.

Parte del placer proporcionado por los ordenadores, una vez que ha pasado la novedad de los juegos, proviene del desarrollo de los propios programas y juegos, de hacer que el ordenador lleve a cabo lo que uno quiere con el menor número de instrucciones y necesidades de memoria. Para conseguir esto, usted necesita técnicas que le permitan generar y manipular datos eficazmente, así como poder almacenarlos y recuperarlos con rapidez. Una buena presentación visual de los resultados mejora considerablemente el programa; los listados deberían estar bien documentados y ser fáciles de seguir.

Éste es el objetivo del libro, que contiene alrededor de cien subrutinas para ayudarle a mejorar sus programas. Se han hecho todos los esfuerzos posibles para asegurar que las subrutinas funcionan en los márgenes especificados y de un modo eficaz, pero es posible que haya casos en los que no sea así.

Al final del libro tiene unos cuantos programas completos que ilustran el uso de algunas de las subrutinas. Se trata de dibujos de figuras de cristales, la resolución de un anagrama, el estudio de la estabilidad de un sistema de control automático y la imagen de un triángulo de cualquier tipo, así como un programa verdaderamente eficaz para almacenamiento y recuperación de datos.



Contenido

Las subrutinas

La mayor parte de las subrutinas se puede dividir en tres secciones. La sección central es la subrutina misma, que le ayuda a usted a introducirla como tal y comienza en la línea 1000. La primera sección es una sencilla rutina de entrada que permite comprobar la subrutina una vez ésta ha sido escrita. La sección final empieza siempre en la línea 2000 y es una simple rutina de salida que se usa de la misma forma que la de entrada. Estas tres partes pueden utilizarse en conjunto si usted quiere emplear el programa como un sistema autónomo (como una única rutina).

Cuando en la subrutina misma aparece una sentencia DIM, normalmente ésta es borrada (ERASE) al final de la sección de salida para permitir que se pueda utilizar la subrutina de nuevo. Si la sentencia DIM aparece en la sección de entrada, se supone que forma parte del programa principal y que devolverá la información con RETURN.

Con el fin de facilitar la lectura, se ha adoptado los convenios que explicamos a continuación. Las palabras reservadas y los dos puntos van seguidos de un espacio. Las expresiones matemáticas no llevan espacios; si los argumentos tienen que ir encerrados entre paréntesis, éstos ocupan el lugar del espacio que debe ir tras la palabra reservada. Los números enteros están definidos normalmente, pero los símbolos % y ! no se han empleado en los nombres de las variables de los programas salvo en unos cuantos casos. La práctica totalidad de las cadenas ha sido definida de esta forma, y no usando DEFSTR al principio. Con esto lo que se pretende es hacer el programa más comprensible para los lectores no familiarizados con esta facilidad. SW o SW$ se usan para la información ficticia en operaciones de intercambio de memoria. Los corchetes se emplean en los nombres de matrices.

Hay que tener especial cuidado para impedir que las variables utilizadas entren en conflicto con las del programa principal, ya sea a nivel de escritura, ya a nivel de precisión. A, B, C, H, K, L, M, N, P, Q, R S, T, X, Y y Z son los nombres más comunes.

En cada apartado se da una breve explicación de cómo funciona cada subrutina y, normalmente, se ofrece un ejemplo de una salida típica.

Una vez introducida, la subrutina puede ser comprobada y situada en el lugar preciso con RENUM. De forma alternativa, puede grabar las subrutinas e introducirlas con MERGE donde sean necesarias, siempre que las líneas 1000 en adelante hayan sido reservadas con RENUM.

1. Conversión de números cardinales a ordinales

Esta rutina escribe 'st', 'nd', 'rd' o 'th', según corresponda, a continuación del número que se le entrega. La función definida (FN) escribe A$ si las condiciones son verdaderas, o la cadena vacía si son falsas. La variable BOOLE sólo puede tomar los valores −1 (verdadero) y 0 (falso).

```
10 REM Conversion de cardinal a ordinal (en ingles)
20 DEF FN P$(A$,BOOLE)=MID$(A$,1,-LEN(A$)*BOOLE)
30 N=0: WHILE N<=0 OR N>999999999: INPUT N: WEND
40 IF N<>INT(N) THEN 30
50 N$=MID$(STR$(N),2)
60 GOSUB 1000:END
1000 IF LEN(N$)=1 THEN PRINT N$;FN P$("st",N=1)+FN P$("nd",N=2)+FN
P$("rd",N=3)+FN P$("th",N<>1 AND N<>2 AND N<>3):RETURN
1010 L$=RIGHT$(N$,1): NL$=RIGHT$(N$,2): PRINT N$;FN P$("st",L$="1" AND
NL$<>"11")+FN P$("nd", L$="2" AND NL$<>"12")+FN P$("rd",L$="3" AND
NL$<>"13")+FN P$("th",(L$<>"1" AND L$<>"2" AND L$<>"3" OR
(NL$="11" OR NL$="12" OR NL$="13"))):RETURN
```

2. Rentas y amortizaciones

La primera de estas rutinas calcula la cantidad anual que se recibe cuando se invierte un capital a interés compuesto durante cierto número de años. Por ejemplo, si se invierte un millón a 15 años al 10%, cada año se recibe 131474 pesetas. Esta cantidad es constante a lo largo de los años; parte es interés y parte cantidad. Transcurridos los 15 años, el capital se ha agotado.

La segunda rutina calcula el importe de los pagos mensuales necesarios para amortizar el préstamo. Por ejemplo, si para comprar un piso se pide un préstamo hipotecario de cinco millones al 14%, a devolver en 15 años, cada mes hay que pagar 66587 pesetas. De esta suma, una parte se dedica a amortizar capital y el resto es intereses.

Las operaciones financieras se basan en la serie geométrica. Si I es la tasa de interés (en tanto por ciento), una peseta se convierte en $(1+(I/100))^N$ al cabo de N periodos. Invirtiendo esta fórmula, una peseta de dentro de N años equivale a $1/(1+(I/100))^N$ actuales. Sumando los valores actuales de los pagos futuros se obtiene:

$$A_n = v + v^2 + \ldots + v^n, \text{ donde } v = 1/(1+(I/100))$$

La suma de esta serie es $(1-v^N)/(I/100))$, y es la fórmula utilizada en las subrutinas.

```
10 REM Rentas anuales
20 GOSUB 1000
30 END
```

```
1000 INPUT "Numero de años: ",N
1010 INPUT "Capital invertido: ",C
1020 INPUT "Interes anual (%): ",I
1030 A=(1-(100/(100+I))↑N)/I*100
1040 PRINT USING "#########";C/A;:PRINT" pesetas anuales"
1050 RETURN
```

```
10 REM Amortizaciones mensuales
20 GOSUB 1000
30 END
1000 INPUT "Numero de años: ",N
1010 INPUT "Importe del prestamo: ",C
1020 INPUT "Interes anual (%): ",I:I=I/12
1030 A=(1-(100/(100+I))↑(N*12))/I*100
1040 PRINT USING "######.##";C/A;:PRINT" pesetas mensuales"
1050 RETURN
```

3. La línea mejor ajustada

Esta subrutina calcula la línea recta mejor ajustada a un conjunto de puntos, y traza los puntos y la línea con ejes a escala y etiquetados apropiadamente. También imprime la pendiente y la interceptación de la línea y el coeficiente de correlación para los datos. Las líneas de puntos se dibujan para representar los límites de confianza del 95%, que también se muestran numéricamente.

Las primeras 15 líneas extraen las cantidades numéricas necesarias de los datos introducidos. Dichas cantidades son los valores máximos de x e y (MX,MY), que sirven para hacer el gráfico a una escala conveniente, y los valores medios (XM,YM), a partir de los cuales se puede calcular la pendiente e interceptación de la línea y la varianza.

La línea mejor ajustada viene dada por:

$$y=\frac{\Sigma(x_i-\bar{x})(y_i-\bar{y})}{\Sigma(x_i-\bar{x})^2}\cdot x+\frac{\Sigma y_i}{n}-\frac{\Sigma x_i\cdot\Sigma(x_i-\bar{x})(y_i-\bar{y})}{n\cdot\Sigma(x_i-\bar{x})^2}$$

donde la suma se realiza sobre valores de i desde 1 hasta N.

El coeficiente de regresión se obtiene de

$$r=\frac{\Sigma(x_i-\bar{x})\ (y_i-\bar{y})}{\sqrt{\Sigma(x_i-\bar{x})^2\cdot\Sigma(y_i-\bar{y})^2}}$$

Las seis líneas siguientes a la 1150 permiten encontrar el valor t de Student para calcular los límites de confianza del 95% ±E, lo que depende del número de lecturas así como de la desviación típica.

SCX y SCY son los factores de escala del gráfico, que aseguran que se puede dibujar todos los puntos y que la totalidad de los números aparece en las marcas de los ejes. Si el valor máximo es menor que 1 la escala se incrementa por diez.

La línea 2030 dibuja los ejes, y los cuatro bucles siguientes los indicadores de escala. Los dos bucles posteriores imprimen números en intervalos con cinco subdivisiones, y las líneas que van a continuación escriben los nombres de los ejes.

CHR$(208) se usa para ensanchar los puntos del gráfico; la línea se dibuja mediante la línea 2090. CHR$(171) es el signo ± necesario para los límites de confianza del 95%, que se imprime junto con la pendiente, la interceptación y el coeficiente de correlación.

Finalmente, los valores *X* e *Y* de la línea de puntos son calculados y dibujados mediante SUB 2300.

La subrutina 2300 es la rutina de la línea de puntos, que se explica con más detalle en otra parte de este libro.

```
10 REM Recta mejor ajustada
20 SYMBOL AFTER 208: POKE H,240: POKE H+1,240
30 POKE H+16,255: POKE H+18,255: POKE H+23,0: POKE H+22,0
40 INPUT " Introduzca el numero de pares de lecturas (3 como minimo)";N: IF
N<3 THEN PRINT "No hay bastantes lecturas": GOTO 40
50 DIM E[1,N]
60 PRINT "Escriba los nombres de las variables":PRINT: PRINT
70 PRINT "El nombre de la abscisa es": PRINT: INPUT A$:PRINT
80 PRINT "El nombre de la ordenada es": PRINT :INPUT B$:CLS
90 FOR P=1 TO N
100 LOCATE 1,P: PRINT "x";P;"=";: INPUT E[0,P]
110 LOCATE 20,P: PRINT "y";P;"=";: INPUT E[1,P]
120 NEXT: REM O use DATA INPUT (coordenada x e y) para establecer N y E[1,N]
130 GOSUB 1000: GOSUB 2000: END
1000 MX=0: MY=0: XM=0: YM=0
1010 FOR P=1 TO N
1020 IF E[0,P]>MX THEN MX=E[0,P]
1030 IF E[1,P]>MY THEN MY=E[1,P]
1040 XM=XM+E[0,P]: YM=YM+E[1,P]
1050 NEXT
1060 XM=XM/N: YM=YM/N
1070 XX=0: XY=0: YY=0
1080 FOR P=1 TO N
1090 XX=XX+(E[0,P]-XM)*(E[0,P]-XM)
1100 XY=XY+(E[0,P]-XM)*(E[1,P]-YM)
1110 YY=YY+(E[1,P]-YM)*(E[1,P]-YM)
1120 NEXT
1130 M=XY/XX: C=YM-M*XM
1140 R=XY/(SQR(XX)*SQR(YY))
```

```
1150 DIM T[10]: RESTORE 1150: DATA 12.706,4.103,3.182,2.776,2.571,2.447,
2.365,2.308,2.262,2.228
1160 FOR P=1 TO 10 :READ T[P]: NEXT
1170 IF N<=12 THEN T=T[N-2]
1180 IF N>=13 AND N<=27 THEN T=-0.000014*(N-2)↑3+0.00152*(N-2)
↑2-0.05075*(N-2)+2.5975
1190 IF N>=28 AND N<=62 THEN T=2.056-(N-2)*0.00165
1200 IF N>62 THEN T=1.98
1210 SCX=INT(MX/50)+1: SCY=INT(MY/40)+1
1220 IF MX<=1 THEN SCX=SCX/10
1230 IF MY<=1 THEN SCY=SCY/10
1240 E=T*SQR(1-R*R)*SQR(YY/(N-2))
1250 ERASE T: RETURN
2000 MODE 2: ORIGIN 0,0: CLS
2010 TAG: X=260: Y=385: FOR P=1 TO 20: MOVE X,Y: PRINT MID$("LINEA
MEJOR AJUSTADA",P,1);: X=X+12: NEXT
2020 X=260: Y=365: FOR P=1 TO 20: MOVE X,Y: PRINT CHR$(210);:
X=X+12: NEXT
2030 PLOT 600,70: DRAWR -500,0: DRAWR 0,300
2040 FOR P=200 TO 600 STEP 100: PLOT P,70: DRAWR 0,8: NEXT
2050 FOR P=110 TO 600 STEP 10: PLOT P,70: DRAWR 0,5: NEXT
2060 FOR P=145 TO 370 STEP 75: PLOT 100,P: DRAWR 8,0: NEXT
2070 FOR P=70 TO 370 STEP 7.5: PLOT 100,P: DRAWR 5,0: NEXT
2080 TAG
2090 FOR P=1 TO N: MOVE 96+E[0,P]*10/SCX,74+E[1,P]*7.5/SCY:PRINT
CHR$(208);: NEXT
2100 FOR P=200 TO 600 STEP 100: MOVE P-15,62: PRINT (P-100)*SCX/10;:
NEXT
2110 FOR P=145 TO 370 STEP 75: B=1: IF (P-70)*SCY/7.5<=99 THEN B=2:
IF (P-70)*SCY/7.5<=9 THEN B=3
2120 MOVE 42+10*B,P+8: PRINT (P-70)*SCY/7.5;:NEXT
2130 X=350-5*LEN(A$): Y=38
2140 FOR P=1 TO LEN(A$): Z$=MID$(A$,P,1)
2150 MOVE X,Y: PRINT Z$;
2160 X=X+14+8*(Z$=" "): NEXT
2170 FOR P=1 TO LEN(B$): MOVE 30,230+10*LEN(B$)-20*P: PRINT
MID$(B$,P,1);: NEXT
2180 ORIGIN 100,70
2190 PLOT 0,C*7.5/SCY
2200 DRAWR MX*10/SCX,(MX*M)*7.5/SCY
2210 X1=0: Y1=(E+C)*7.5/SCY: X2=MX*10/SCX:
Y2=(MX*M+E+C)*7.5/SCY
2220 GOSUB 2300
2230 X1=0: Y1=(-E+C)*7.5/SCY: X2=MX*10/SCX:
Y2=(MX*M-E+C)*7.5/SCY
2240 GOSUB 2300
```

```
2250 MOVE 20,300: PRINT "M=";M;: MOVE 350,300: PRINT "C=";C;
2260 MOVE 320,50: PRINT "R=";R;: MOVE 250,25: PRINT "95% Conf
y=";CHR$(32);E;: MOVE 344,25: PRINT CHR$(171);
2270 TAGOFF: ORIGIN 0,0
2280 IF INKEY$="" THEN 2280
2290 MODE 1: RETURN
2300 IF X1=X2 THEN 2370 ELSE IF Y1=Y2 THEN 2330
2310 P=(Y2-Y1)/(X2-X1)
2320 IF ABS(P)<1 THEN 2340 ELSE P=1/P: GOTO 2380
2330 P=0
2340 I=10*SGN(X2-X1)/SQR(1+P*P)
2350 FOR Q=0 TO (X2-X1)/I
2360 PLOT X1+Q*I,Y1+Q*I*P: DRAWR 0.4*I,0.4*I*P: NEXT: RETURN
2370 P=0
2380 I=10*SGN(Y2-Y1)/SQR(1+P*P)
2390 FOR Q=0 TO (Y2-Y1)/I
2400 PLOT X1+Q*I*P,Y1+Q*I: DRAWR 0.4*I*P,0.4*I: NEXT: RETURN
2410 RETURN
```

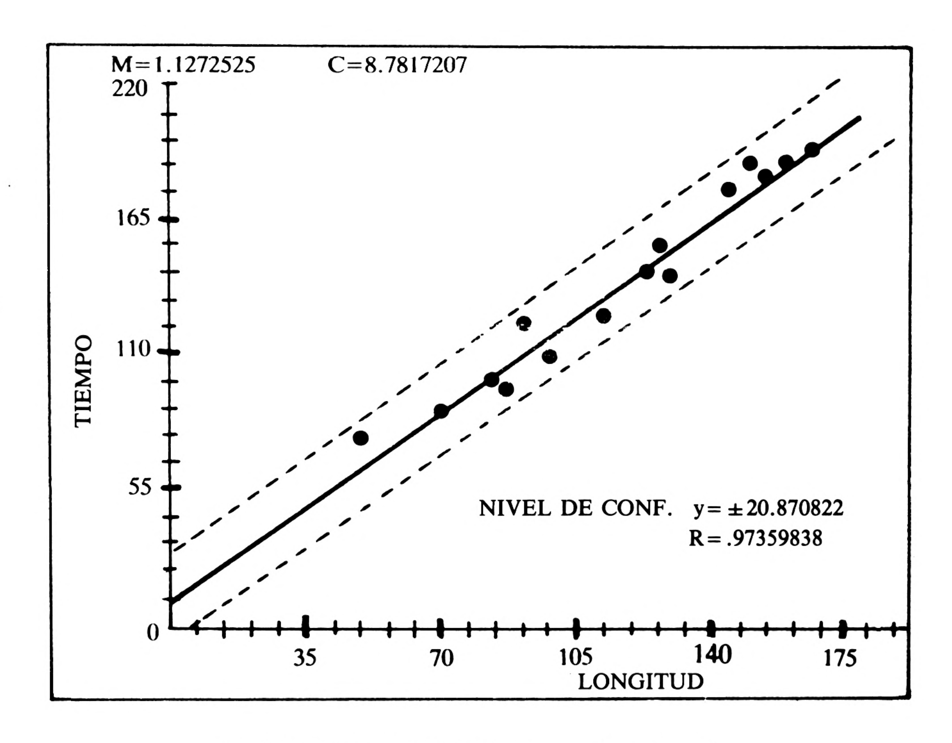


Fig. 3.1. Presentación de la línea mejor ajustada.

4. Árbol de búsqueda binario

Un «árbol de búsqueda binario» es una útil forma de estructura de almacenamiento que permite un acceso rápido a los datos. Respecto a la terminología del árbol, véase el diagrama. Los vértices se designan con A[V], un elemento del conjunto. Cada vértice U de un subárbol situado a la izquierda tiene la designación A[U]>A[V], y los de un subárbol de la derecha la designación A[U]<A[V]. Sólo hay un vértice para cada valor del conjunto. A[V]=O implica un vértice vacío.

Esta subrutina almacena N valores que deben ser diferentes; el cero no está permitido. También se puede manejar cadenas cambiando A[M] por A$[M]. (Z se transformaría en Z$, 0 en "", etc.)

La línea 50 encuentra el tamaño del árbol requerido. Si un subárbol de éste es muy largo, por ejemplo porque los valores están colocados por orden, quizá sea necesario incrementar el valor inicial de K (aparecería el mensaje Subscript out of range, 'subíndice fuera del margen').

Al tiempo que se introducen los valores, el programa avanza desde la raíz hacia las hojas averiguando:

1. Si un vértice está ya ocupado.
2. Si lo está, ¿es menor el valor? Si es así, va a la izquierda;
3. de lo contrario, va a la derecha.
4. Cuando encuentra un vértice vacío lo designa con el valor.
5. Siguiente valor (NEXT).

La rutina de búsqueda sigue la misma pauta. K1 es la altura del vértice dentro del árbol.

Véase la secuencia INORDER como rutina relacionada con ésta.

```
10 REM Arbol de busqueda binario
20 DEFINT N,K,P,V
30 N=0: WHILE N<=0: INPUT "Numero de valores";N: WEND
40 M=N+1: K=2
50 WHILE M>1: M=M/2: K=K+1: WEND
60 M=2↑K: DIM A[M]: GOSUB 1000: GOSUB 2000: GOTO 1050
1000 INPUT "Valor medio";A[1]
1010 FOR P=2 TO N:PRINT  "Valor";P;: INPUT Z: V=1
1020 IF Z>A[V] THEN V=V+V+1: IF A[V]<>0 THEN 1020
ELSE A[V]=Z:GOTO 1040
1030 V=V+V: IF A[V]<>0 THEN 1020 ELSE A[V]=Z
1040 NEXT: RETURN
1050 REM Rutina de busqueda
1060 PRINT: INPUT "Lo siguiente es un miembro";Z
1070 V=1: K1=1
1080 IF Z=A[V] THEN PRINT Z;" es un miembro": GOTO 1050
```

```
1090 IF K1<K THEN K1=K1+1: IF Z>A[V] THEN V=V+V+1: GOTO 1080 ELSE
V=V+V: GOTO 1080
1100 PRINT Z;"no es un miembro": GOTO 1050
2000 FOR P=1 TO 2↑(K-1)-1: PRINT A[P];SPACE$(7-LEN(STR$(A[P])));: NEXT
2010 RETURN
```

Ejemplo

```
run                          Resultado
Numero de valores? 25        52  35  60  20  45
Valor medio? 52              59  70  10  28  40
Valor 2?  35                 48  54   0  65  75
Valor 3?  60                  1  15  26   0  37
Valor 4?  20                 41   0  49  53  55
Valor 5?  45                  0   0  63  69  74
Valor 6?  59                  0   0   0   0   0
Valor 7?  70                  0   0   0   0   0
Valor 8?  10                  0   0   0   0   0
Valor 9?  28                  0   0   0   0   0
Valor 10? 40                  0   0   0   0   0
Valor 11? 48                  0   0   0   0   0
Valor 12? 54                  0   0   0
Valor 13? 65                 Ready
Valor 14? 75
Valor 15? 1
Valor 16? 15
Valor 17? 26
Valor 18? 37
Valor 19? 41
Valor 20? 49
Valor 21? 53
Valor 22? 55
Valor 23? 63
Valor 24? 69
Valor 25? 74
```

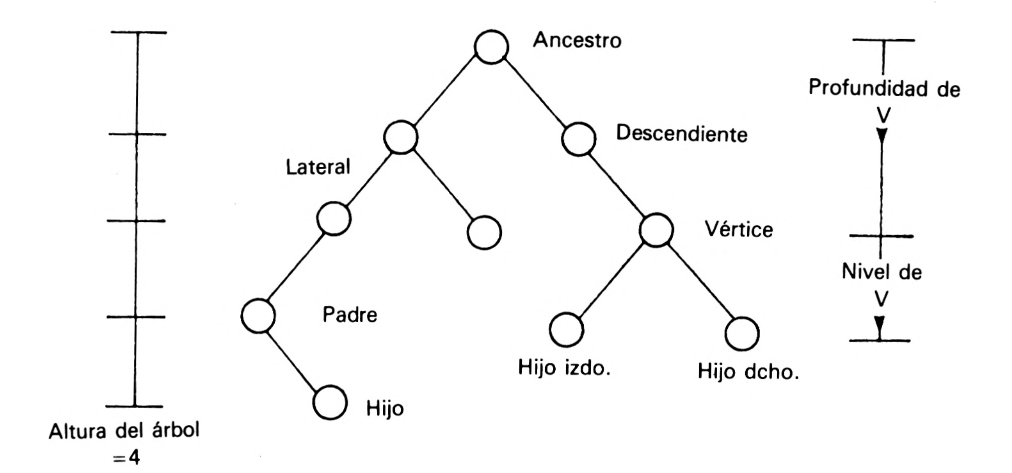


Fig. 4.1a. Árbol de tipo familiar.

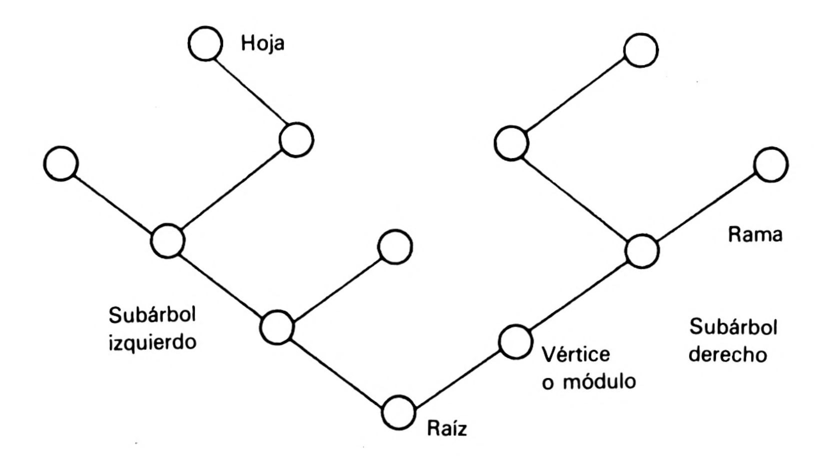


Fig. 4.1b. Árbol de tipo botánico.

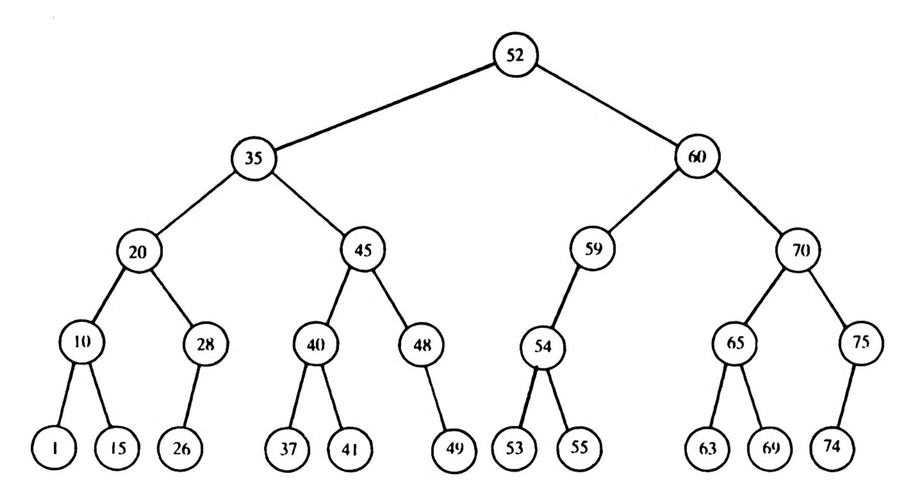


Fig. 4.2. Árbol de búsqueda binario.

5. Coeficientes binomiales

Esta subrutina calcula el coeficiente de $(a+b)^n$ para $n<=124$.

Los coeficientes binomiales son útiles para calcular la probabilidad de ocurrencia de algún suceso. Por ejemplo, la extracción de tres bolas de un saco que contiene tres bolas rojas y siete negras se rige por

$$(3/10+7/10)^3 = 1x(0.3)^3 + 3x(0.3)^2x(0.7) + 3x(0.3)x(0.7)^2 + 1x(0.7)^3$$
$$= 0.027 \quad +0.189 \quad +0.441 \quad +0.343$$

El primer término es la probabilidad de extraer tres bolas rojas; el segundo la de dos bolas rojas y una negra; el tercero, una roja y dos negras; y, finalmente, el último término es la probabilidad de que todas las bolas sean negras.

La ilustración es el triángulo de Pascal, donde todas las filas contienen un conjunto de coeficientes binomiales. Tiene la propiedad de que cada fila puede extraerse de la anterior sumando cada coeficiente con el que tiene a su lado.

Observe cómo las sentencias DIM alternativas usan el valor de T dependiendo de si la serie es finita o infinita.

```
10 REM Coeficientes binomiales
20 INPUT "Valor de n";N:IF ABS(INT(N))<>N THEN PRINT "n no es un entero
positivo. Elija otro numero": INPUT T: GOSUB 1010: GOTO 40
30 GOSUB 1000
40 GOSUB 2000: END
1000 DIM B[N+1]: T=N+1: GOTO 1020
```

```
1010 DIM B[T]
1020 Z=1: M=N
1030 FOR P=1 TO T: B[P]=Z: Z=Z*M/P: M=M-1: NEXT
1040 RETURN
2000 FOR P=1 TO T: PRINT B[P]: NEXT
2010 ERASE B: RETURN
```

Triángulo de Pascal

```
10 REM Triangulo de Pascal
20 DEFINT H,P,Q
30 H=0: WHILE H<=0: INPUT "Altura (maximo 9)";H: WEND
40 DIM B[H,H+1]
50 FOR P=0 TO H: Z=1: M=P
60 FOR Q=1 TO P+1: B[P,Q]=Z
70 Z=Z*M/Q: M=M-1
80 NEXT Q,P
90 CLS: PRINT :PRINT
100 FOR P=0 TO H: PRINT SPACE$(18-P*2);
110 FOR Q=1 TO P+1: A$=STR$(B[P,Q])
120 PRINT SPACE$(4-LEN(A$));A$;: NEXT Q: PRINT
130 PRINT: NEXT P
```

```
                  1
                1   1
              1   2   1
            1   3   3   1
          1   4   6   4   1
        1   5  10  10   5   1
      1   6  15  20  15   6   1
    1   7  21  35  35  21   7   1
  1   8  28  56  70  56  28   8   1
1   9  36  84 126 126  84  36   9   1
```

6. Circunferencia

Estas subrutinas se han empleado para calcular las posiciones de centros de circunferencias en dibujos de ingeniería y para aproximar figuras complejas a una serie de arcos para el funcionamiento de una máquina NC.

La primera subrutina calcula las coordenadas del centro y el radio de una única circunferencia que pasa por tres puntos. Observe que, al tener una raíz cuadrada dos valores, *R* puede ser positivo o negativo. Por tanto, debe usted tener cuidado si lo usa en una fórmula como valor absoluto.

La segunda subrutina calcula la posición de los centros de dos circunferencias de un diámetro dado y que pasan por dos puntos.

```
10 REM Busqueda del centro del circulo
20 CLS: PRINT "   Introduzca las coordenadas de los tres puntos": PRINT
30 INPUT "x1=";X1: LOCATE 20,4: INPUT "y1=";Y1
40 INPUT "x2=";X2: LOCATE 20,5: INPUT "y2=";Y2
50 INPUT "x3=";X3: LOCATE 20,6: INPUT "y3=";Y3
60 GOSUB 1000: GOSUB 2000: END
1000 D=X1*(Y2-Y3)+X2*(Y3-Y1)+X3*(Y1-Y2): IF ABS(D)<0.000001 THEN
GOSUB 2050: GOTO 20
1010 OX=((X1*X1+Y1*Y1)*(Y2-Y3)+(X2*X2+Y2*Y2)*(Y3-Y1)+
(X3*X3+Y3*Y3)*(Y1-Y2))/2/D
1020 OY=((X1*X1+Y1*Y1)*(X3-X2)+(X2*X2+Y2*Y2)*(X1-X3)+
(X3*X3+Y3*Y3)*(X2-X1))/2/D
1030 R=SQR((OX-X1)*(OX-X1)+(OY-Y1)*(OY-Y1))
1040 RETURN
2000 PRINT: PRINT "Radio=";R
2010 PRINT: PRINT "Coordenadas del centro del area"
2020 PRINT: PRINT "Ox=";OX
2030 PRINT: PRINT "Oy=";OY
2040 RETURN
2050 PRINT "    Error en los datos. Pulse otra tecla para volver a empezar"
2060 IF INKEY$="" THEN 2060
2070 RETURN
```

```
10 REM Busqueda de los centros de dos circulos
20 CLS: PRINT "Introduzca las coordenadas de los dos puntos": PRINT
30 INPUT "x1=";X1: LOCATE 20,4: INPUT "y1=";Y1: PRINT
40 INPUT "x2=";X2: LOCATE 20,6: INPUT "y2=";Y2: PRINT
50 PRINT "Introduzca el radio del circulo": PRINT
60 INPUT "r=";R
70 GOSUB 1000: END
1000 DEF FN R(A)=A*-(ABS(A)>0.000001): NL$=CHR$(10)+CHR$(13)
1010 IF (X1-X2)*(X1-X2)+(Y1-Y2)*(Y1-Y2)>4*R*R THEN PRINT "Error de
datos.  Pulse otra tecla para volver a empezar":GOSUB 1120: RUN
1020 IF ABS(X2-X1)<0.000001 AND ABS(Y2-Y1)<0.000001 THEN PRINT "Arco
demasiado pequeño para calcular los centros.  Pulse otra tecla":GOSUB 1120:
RUN
1030 IF ABS(Y2-Y1)<0.000001 THEN THETA=PI/2: GOTO 1050
1040 THETA=ATN((X1-X2)/(Y1-Y2))
1050 XM=(X1+X2)/2: YM=(Y1+Y2)/2
1060 P=SQR(R*R-(Y1-YM)*(Y1-YM)-(X1-XM)*(X1-XM))
1070 PC=P*COS(THETA): PS=P*SIN(THETA)
1080 PRINT: PRINT
```

```
1090 PRINT "Centros de las areas";NL$;FN R(XM+PC);",";FN R(YM+PS)
1100 PRINT "y";NL$;FN R(XM-PC);",";FN R(YM-PS)
1110 RETURN
1120 IF INKEY$="" THEN 1120
1130 RETURN
```

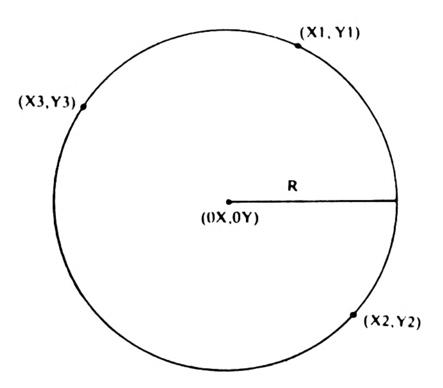


Fig. 6.1. Circunferencia que atraviesa Ttres puntos.

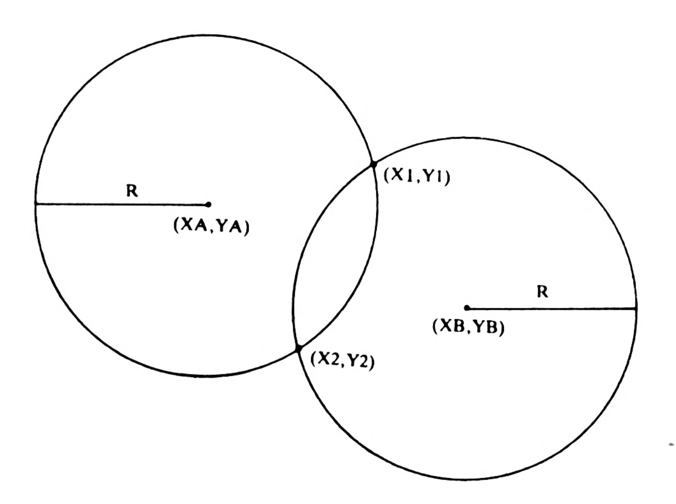


Fig. 6.2. Circunferencia que atraviesa dos puntos

7. Combinaciones de +1 y −1 en grupos de tres

Esta subrutina fue desarrollada para el programa del cristal cúbico y produce las ocho combinaciones de +1 y −1 en grupos de tres, no habiendo ninguna diferencia entre la primera de dichas combinaciones y la última de ellas. Los elementos de cada elección son dos, +1 y −1, y, por tanto, el número total de disposiciones es $2\times2\times2=8$.

La primera combinación es

1 1 1

El procedimiento es intercambiar dos de los números o multiplicar a uno de ellos por −1, lo que da la siguiente secuencia de cambios:

A=0	1	1	1	A(2)∗−1
1	1	1	−1	A(1)↔A(2)
2	1	−1	1	A(2)∗−1
3	1	−1	−1	A(0)∗−1
4	−1	−1	−1	A(2)∗−1
5	−1	−1	1	A(1)↔A(2)
6	−1	1	−1	A(2)∗−1
7	−1	1	1	A(0)∗−1
8	1	1	1	

```
10 REM Combinaciones de +1 y −1 en grupos de tres
20 GOSUB 1000: GOSUB 2000: END
1000 DIM A[2],C[8,2]
1010 A[0]=1: A[1]=1: A[2]=1
1020 FOR A=0 TO 8: FOR B=0 TO 2
1030 C[A,B]=A[B]: NEXT
1040 IF A=1 OR A=5 THEN Z=A[1]: A[1]=A[2]:A[2]=Z: GOTO 1070
1050 IF A=3 OR A=7 THEN A[0]=−A[0]: GOTO 1070
1060 A[2]=−A[2]
1070 NEXT
1080 RETURN
2000 FOR A=0 TO 8: FOR B=0 TO 2
2010 PRINT C[A,B];: NEXT
2020 PRINT: NEXT:
2030 ERASE A,C
2040 RETURN
```

8. Números complejos

Los números complejos del tipo $a+bi$ (donde i es la raíz cuadrada de -1) son un medio muy útil de representar pares de números. En términos geométricos, a es la coordenada x y b la coordenada y; por otra parte, los números complejos son bastante frecuentes en matemáticas.

A primera vista, podría parecer que el ordenador no reconoce el tratamiento de los números complejos como SQR(−1). Sin embargo, si se almacena por separado, la parte real y la imaginaria de un número complejo en dos matrices, el manejo de estos números no es más difícil que el de los ordinarios. Es necesario usar una orden PRINT para incorporar la i a la respuesta final.

Estas subrutinas manejan operaciones sencillas como la suma, resta, multiplicación y división, pero puede encontrar operaciones más complicadas en el apartado 'Matrices'.

Las fórmulas empleadas son las siguientes:

$(a+bi)$	más	$(c+di)=(a+c)+(b+d)i$
$(a+bi)$	menos	$(c+di)=(a-c)+(b-d)i$
$(a+bi)$	por	$(c+di)=(ac-bd)+(bc+ad)i$
$(a+bi)$	dividido por	$(c+di)=(ac+bd)/(c^2+d^2)+(bc-ad)/(c^2+d^2)i$

El DEF FN para la última sentencia PRINT permite respetar los convenios matemáticos normales y evitar expresiones recargadas, como por ejemplo

$0+-2i$, es decir, $-2i$, y $-2+0i$, es decir, -2

```
10 REM Numeros complejos
20 DEF FN A$(A$,A,BOOLE)=MID$(A$,1+A,-LEN(A$)*BOOLE)
30 PRINT "Primer numero complejo Z1=a+ib"
40 PRINT: INPUT "a=";A: INPUT "b=";B
50 PRINT "Segundo numero complejo Z2=c+id"
60 PRINT :INPUT "c=";C: INPUT "d=";D
70 CLS: GOSUB 2000: END
2000 R=A: I=B: GOSUB 2100: PRINT: PRINT "Z1=";Z$
2010 R=C: I=D: GOSUB 2100: PRINT: PRINT "Z2=";Z$
2020 R=A+C: I=B+D: GOSUB 2100: PRINT: PRINT "Z1+Z2=";Z$
2030 R=A-C: I=B-D: GOSUB 2100: PRINT: PRINT "Z1-Z2=";Z$
2040 R=A*C-B*D: I=B*C+A*D: GOSUB 2100: PRINT: PRINT "Z1*Z2=";Z$
2050 S=C*C+D*D: IF S=0 THEN GOTO 2080
2060 R=(A*C+B*D)/S: I=(B*C-A*D)/S: GOSUB 2100: PRINT: PRINT"
Z1/Z2="Z$
2070 PRINT: RETURN
2080 PRINT: PRINT "Denominador cero, respuesta infinita para la division"
2090 PRINT: RETURN
```

```
2100 Z$=FN A$("-",0,R<0)+FN A$(STR$(R),1,R<>0)+FN A$("+",0,ABS(R)
>0 AND I>0)+FN A$("-",0,I<0)+FN A$(STR$(I),1,I<>0 AND ABS(I)<>1)
+FN A$("i",0,I<>0)+FN A$("0",0,R=0 AND I=0)
2110 RETURN
```

Ejemplo

```
run
Primer numero complejo Z1=a+bi

a=? 3
b=? -1

Segundo numero complejo Z2=c+di

c=? -3
d=? 4

Z1=3-i
Z2=-3+4i
Z1+Z2=3i
Z1-Z2+6-5i
Z1*Z2=-5+15i
Z1/Z2=-.52-.36i

Ready
```

9. Filtro de números

Esta subrutina de filtro sólo admite enteros positivos y negativos y números decimales o exponenciales. El número se introduce en forma de cadena y es comprobado carácter por carácter.

La línea 1000 inicializa las variables. Q es la posición del punto decimal en la mantisa y R la posición de 'E' o 'e', si es que están presentes. S es la posición de cualquier '+' o '−' detrás del exponente. W se emplea para comprobar que hay un carácter tras la posición de '+' o '−'. FAIL se usa para el regreso (RETURN) tras las comprobaciones. Todas estas variables están establecidas con valor cero.

T tiene inicialmente el valor 1, pero cambia a 2 si se usa '+' o '−' delante del número de forma que el signo no esté incluido en las comprobaciones.

La línea 1040 busca una forma de número exponencial y continúa la comprobación en la línea 1110.

La línea 1050 comprueba si hay un decimal en números no exponenciales, y la 1060 rechaza entradas tales como "+.", "+", "." y las menores que 10↑38 o mayores que 10↑−38.

La línea 1100 rechaza los números exponenciales que no tienen mantisa. Las líneas siguientes buscan un punto decimal en la mantisa y excluyen los números exponenciales sin exponente o los que tienen un punto decimal en el mismo. Se encuentra la posición detrás del exponente de los signos "+" o "−", y los números sin entrada después del signo son rechazados.

De vuelta a la línea 1110, se comprueban las características numéricas de todos los caracteres, excepto de los "+" o "−" iniciales, la "E" o "e", el punto decimal y el exponente "+" o "−".

Finalmente, se comprueba el valor numérico del exponente para asegurar que está entre 10↑38 y 10↑−38, habiendo trabajado primero con todos los números con una mantisa igual a cero (debido al LOG requerido en la prueba).

```
10 REM Filtro de numeros
20 INPUT "Numero";Q$: GOSUB 1000
30 IF FAIL THEN 20 ELSE GOSUB 2000: END
1000 Q=0: R=0: S=0: FAIL=0: T=1: L=LEN(Q$)
1010 IF Q$="" THEN GOSUB 1170 : RETURN
1020 IF LEFT$(Q$,1)="-" OR LEFT$(Q$,1)="+" THEN T=2
1030 FOR P=T TO L: Z$=MID$(Q$,P,1)
1040 IF Z$="E" OR Z$="e" THEN 1110 ELSE NEXT P
1050 FOR P=T TO L: IF MID$(Q$,P,1)="." THEN Q=P
1060 NEXT P: IF Q=T AND L>38 OR L-Q>38 OR L<T OR L<Q+1 THEN
GOSUB 1170: RETURN
1070 FOR P=T TO L: IF P<>Q AND P<>R AND P<>S THEN IF MID$(Q$,P,1)
<"0" OR MID$(Q$,P,1)>"9" THEN GOSUB 1170: RETURN
1080 NEXT P: IF R<>0 THEN GOSUB 1180
1090 RETURN
1100 IF P<=T THEN GOSUB 1170: RETURN
1110 R=P: FOR P=T TO R-1: IF MID$(Q$,P,1)="." THEN Q=P
1120 NEXT P: IF L=R OR R<=Q+1 THEN GOSUB 1170: RETURN
1130 IF MID$(Q$,R+1,1)="+" OR MID$(Q$,R+1,1)="-" THEN S=R+1: W=1
1140 IF L<=R+W THEN GOSUB 1170: RETURN
1150 GOTO 1070
1170 FAIL=-1:RETURN
1180 IF VAL(MID$(Q$,T,R-T))=0 THEN Q$="0": RETURN
1190 V=VAL(MID$(Q$,R+1))+LOG10(VAL(MID$(Q$,T,R-T))): IF ABS(V)>=38
THEN GOSUB 1170: RETURN
1200 RETURN
2000 N=VAL(Q$): PRINT N: RETURN
```

10. Paréntesis condicionales

Esta subrutina coloca paréntesis en torno a ciertos elementos al ser éstos impresos de acuerdo con el resultado de una operación lógica. Por ejemplo, en los cálculos a veces se usan más los paréntesis que la tinta roja o el signo menos para indicar una entrada negativa. La variable BOOLE sería entonces ($N<0$).

La rutina funciona cambiando el código de la cadena de caracteres de 32 (blanco) a 40 o 41, los paréntesis de abrir y cerrar respectivamente. Esta técnica se puede usar para otros caracteres, y se emplea en el apartado 'Determinante por el desarrollo de Laplace' para transformar el signo "+" en "−" en la expresión literal.

```
10 REM Parentesis condicionales
20 DEF FN L$(BOOLE)=CHR$(32-8*BOOLE)
30 DEF FN R$(BOOLE)=CHR$(32-9*BOOLE)
35 REM La variable de BOOLE solo toma los valores 0 (Falso) o -1 (Verdadero)
40 DEF FN P$(A)=MID$(STR$(A),2)
50 GOSUB 1000: END
1000 FOR A=1 TO 4: FOR B=1 TO 5
1010 PRINT FN L$(A<B);FN P$(A-B);FN R$(A<B),A-B
1020 NEXT B,A: RETURN
```

11-16. Conversiones

En los ordenadores Amstrad son posibles las conversiones entre números binarios, decimales y hexadecimales, pero están limitadas a dieciséis bits. Las siguientes rutinas se pueden emplear para enteros positivos de cualquier tamaño.

11. Conversión de binario a decimal

Esta rutina encuentra el número decimal sumando el número de los pesos 1, 2, 4, etc.

```
10 REM Conversion de binario a decimal
20 REM Utilice la rutina de comprobacion de numeros binarios
30 INPUT "Introduzca numero binario";B$
40 GOSUB 1000: GOSUB 2000: END
1000 D=0: C=1: L=LEN(B$)
1010 D=D+2↑(L-C)*VAL(MID$(B$,C,1))
1020 IF L=C THEN RETURN ELSE C=C+1: GOTO 1010
1030 RETURN
2000 PRINT B$;SPC(5);D: RETURN
```

12. Conversión de binario a hexadecimal

Entre los números binarios y hexadecimales hay una relación sencilla, ya que en un número binario cada grupo de cuatro representa un número hexadecimal. La función B$(S) es la lista de 16 dígitos hexadecimales, de los cuales se puede seleccionar el correcto mediante S, que está formado por cuatro ciclos de Q y termina cuando T=L.

```
10 REM Conversion de binario a hexadecimal
20 DEF FN B$(S)=MID$("0123456789ABCDEF",S,1)
30 LINE INPUT "Numero binario ";B$
40 REM Utilice la rutina de comprobacion de numeros binarios
50 GOSUB 1000: GOSUB 2000: END
1000 L=LEN(B$): H$=STRING$(1+INT((L-1)/4),32): T=0
1010 S=0: Q=0
1020 S=S+2↑Q*VAL(MID$(B$,L-Q-INT(T/4)*4,1)): Q=Q+1: T=T+1
1030 IF T=L THEN MID$(H$,1,1)=FN B$(S+1): RETURN
1040 IF Q=4 THEN MID$(H$,LEN(H$)-T/4+1)=FN B$(S+1): GOTO 1010 ELSE
GOTO 1020
2000 PRINT B$;SPC(5);H$
2010 RETURN
```

13. Conversión de decimal a binario

Esta rutina se basa en el resto de la división sucesiva por 2. Como en los cálculos de ordenador hay errores de redondeo, se suma 0.0000001 para dar el valor correcto de INT.

```
10 REM Conversion de decimal a binario
20 DEFINT P,Q
30 INPUT "Introduzca numero decimal";D
40 GOSUB 1000: GOSUB 2000: END
1000 P=1: WHILE 2↑P<ABS(D)+1: P=P+1: WEND
1010 A$=STRING$(P,32): FOR Q=1 TO P
1020 U=INT(D/2↑Q-INT(D/2↑Q+0.0000001)+0.5)
1030 MID$(A$,1+P-Q)=RIGHT$(STR$(U),1): NEXT
1040 RETURN
2000 PRINT D;SPC(5);A$
2010 RETURN
```

14. Conversión de decimal a hexadecimal

Esta rutina parte la lista hexadecimal en FN H$(H) con los restos de la división por potencias de 16.

```
10 REM Conversion de decimal a hexadecimal
20 DEF FN H$(H)=MID$("0123456789ABCDEF",1+16*(H/16-INT(H/16)),1)
30 INPUT "Introduzca numero decimal";N: IF N<0 THEN 30
40 GOSUB 1000: GOSUB 2000: END
1000 IF N=0 THEN A$="0": RETURN
1010 A$=""
1020 FOR D=LEN(STR$(N))-2 TO 0 STEP -1
1030 A$=A$+FN H$(INT(N/(16↑D))): NEXT
1040 IF LEFT$(A$,1)="0" THEN A$=RIGHT$(A$,LEN(A$)-1): GOTO 1040
1050 RETURN
2000 PRINT N;SPC(5);A$
2010 RETURN
```

15A. Conversión de hexadecimal a binario (1)

La función H(A) convierte H$ en un número basándose en el hecho de que los dígitos tienen códigos ASCII del 48 al 57, y las mayúsculas A-F del 65 al 70. Por ejemplo, ASC(C) es 67; reste 48 y 7 ("C">"A", por tanto el paréntesis es igual a −7), lo que da 12 (el valor hexadecimal de C).

```
10 REM Conversion de hexadecimal a binario (1).
20 DEF FN H(A)=ASC(MID$(H$,A,1))-48+7*(MID$(H$,A,1)>="A")
30 LINE INPUT "NUMERO HEX. ";H$
40 GOSUB 1000: GOSUB 2000: END
1000 B$=STRING$(LEN(H$)*4,32)
1010 FOR N=LEN(H$) TO 1 STEP -1: FOR M=1 TO 4
1020 U=INT(FN H(N)/2↑M-INT(FN H(N)/2↑M)+0.5000005)
1030 MID$(B$,4*N-M+1)=RIGHT$(STR$(U),1): NEXT M,N
1040 RETURN
2000 FOR T=0 TO 2: IF MID$(B$,1+T,1)="0" THEN NEXT T
2010 PRINT H$;SPC(5);RIGHT$(B$,LEN(B$)-T)
2020 RETURN
```

15B. Conversión de hexadecimal a binario (2)

La función H(A) descifra el número hexadecimal usando el orden de letras y números de la lista ASCII y extrayendo caracteres con ASC(MID$(H$,A,1))−47. Por ejemplo, ASC(F)=70, así que se selecciona el 23.º carácter de la cadena, es decir, el "?", que tiene un valor ASC de 63. Reste 48 y esto dará 15 (el valor decimal de F).

```
10 REM Conversion de hexadecimal a binario (2). Mayusculas o minusculas
20 DEF FN H(A)=ASC(MID$("0123456789*******:;<=>?************
**************:;<=>?",ASC(MID$(H$,A,1))-47,1))-48
30 LINE INPUT "H$ (MAYUSCULAS o minusculas) ";H$
40 GOSUB 1000: GOSUB 2000: END
1000 B$=STRING$(LEN(H$)*4,32)
1010 FOR N=LEN(H$) TO 1 STEP -1: FOR M=1 TO 4
1020 U=INT(FN H(N)/2↑M-INT(FN H(N)/2↑M)+0.5000005)
1030 MID$(B$,4*N-M+1)=RIGHT$(STR$(U),1): NEXT M,N
1040 RETURN
2000 FOR T=0 TO 2: IF MID$(B$,T+1,1)="0" THEN NEXT T
2010 PRINT H$;SPC(5);RIGHT$(B$,LEN(B$)-T)
2020 RETURN
```

La división sucesiva desemboca en U, que se incorpora a B$ para formar el número binario. El bucle de T suprime los ceros superfluos de la parte delantera del número binario.

16A. Conversión de hexadecimal a decimal (1)

La función H(A) convierte H$ de la misma forma que lo hacía la rutina de conversión de hexadecimal a binario. Sólo funciona para las letras mayúsculas debido al UPPER$ de la línea 30.

```
10 REM Conversion de hexadecimal a decimal (1).
20 DEF FN H(A)=ASC(MID$(H$,A,1))-48+7*(MID$(H$,A,1)>="A")
30 LINE INPUT "H$ ";H$: H$=UPPER$(H$)
40 GOSUB 1000: GOSUB 2000: END
1000 D=0: FOR N=1 TO LEN(H$)
1010 D=D+FN H(N)*16↑(LEN(H$)-N): NEXT: RETURN
2000 PRINT H$;SPC(5);D
2010 RETURN
```

16B. Conversión de hexadecimal a decimal (2)

Esta función, más complicada, se usa como la anterior, pero para trabajar con letras mayúsculas y minúsculas en el número hexadecimal.

```
10 REM Conversion de hexadecimal a decimal (2).
20 DEF FN H(A)=ASC(MID$("0123456789*******:;<=>?************
**************:;<=>?",ASC(MID$(H$,A,1))-47,1))-48
30 LINE INPUT "Numero hex., MAYUSCULA o minuscula ";H$
40 GOSUB 1000: GOSUB 2000: END
1000 D=0: FOR N=1 TO LEN(H$)
1010 D=D+FN H(N)*16↑(LEN(H$)-N): NEXT
1020 RETURN
2000 PRINT H$;SPC(5);D
2010 RETURN
```

17-20. Comprobación de la entrada de datos

El objetivo de una rutina de entrada de datos es doble. En primer lugar, debería permitir que la entrada fuera comprobada automáticamente para asegurar que pertenece al tipo apropiado y que se encuentra dentro de los márgenes correctos, es decir, que es un número, un nombre, letras sueltas, etc. En segundo lugar, debería hacer posible que el operador del teclado comprobara la entrada para asegurarse de que los valores, la escritura, etc. son correctos antes de usarlos. Los datos no comprobados llevarán inevitablemente a que el programa falle alguna vez. Por ejemplo, INPUT N: N=1/N desembocará en la división por cero si se pulsa accidentalmente la tecla INTRO antes que la tecla del dígito. Esto se evitaría sencillamente con A INPUT N: IF N=0 THEN A.

Otras comprobaciones subsanan este error. Por ejemplo, IF MID$(N$,P,1)>="0" AND MIS$(N$,P,1)<="9" (en un bucle) THEN 'aceptar' asegura que todos los caracteres de un número son dígitos. Si usted está estableciendo una sentencia DIM, asegúrese de que no puede ser negativa, es decir, IF VAL(N$)<0 THEN (INPUT número de línea).

Una forma generalizada de entrada (INPUT) es

número de línea LET Q =(*Número de línea*): INPUT Q$

Q$ puede ser comprobada carácter a carácter con un bucle, y, si falla, va a Q (THEN GOTO Q). Si no, Q$ es convertida en un número mediante VAL y almacenada, o bien es alamacenada directamente, permaneciendo Q y Q$ disponibles para la siguiente entrada.

17. Entrada de datos (ecuaciones lineales con hasta ocho variables)

Esta subrutina acepta coeficientes y constantes (un máximo de ocho) para conjuntos de ecuaciones antes de resolverlas en un sistema, por ejemplo.

La limitación a 8 variables proviene del tamaño de la pantalla. FN BS$ tiene las funciones de borrado de la pantalla y retroceso de un espacio, y FN A$ le da a la impresión un formato empaquetado. FN K y FN J evitan complicaciones en las posiciones de PRINT.

La primera mitad de la rutina imprime las ecuaciones en forma algebráica, concatenando CHR$ (96+K) con FN A$(J), "*" y CHR$(90+K−N) para formar a1*x,b2*y etc. En la línea 1090 se insertan los valores numéricos y a1 se sustituye por un blanco, la posición PRINT retrocede y se escribe E[J,K] donde antes estaba a1*x.

Si no está usted satisfecho con los valores numéricos introducidos, puede volver a la línea 1090 y cambiarlos mediante K$.

```
10 REM Introduccion de datos (ecuaciones lineales)
20 SYMBOL AFTER 208: POKE HIMEN+1,0
30 DEFINT J,K,N
40 DEF FN BS$(A)=SPACE$(A)+STRING$(A,8)
50 DEF FN A$(A)=MID$(STR$(A),2)
60 DEF FN K(K)=8*K+32*(K>4)-6: DEF FN J(J)=2*J-(K>4)+4
70 INPUT "Cuantas variables hay";N: IF N<1 OR N>8 THEN 70
75 DIM E[N,N+1]
80 GOSUB 1000: END
1000 CLS: LOCATE 10,2: PRINT "Sus ecuaciones son"
1010 LOCATE 10,3: FOR P=1 TO 18: PRINT CHR$(208);: NEXT
1020 FOR J=1 TO N: FOR K=1 TO N
1030 LOCATE FN K(K),FN J(J): PRINT CHR$(96+K)+FN A$(J)+"*"+CHR$
(90+K-N): NEXT
1040 FOR K=1 TO N-1
1050 LOCATE FN K(K)+5,FN J(J): PRINT "+": NEXT
1060 LOCATE 30,FN J(J): PRINT " = "+"K"+FN A$(J): NEXT
1070 FOR K=1 TO N
1080 LOCATE 33,4: PRINT "Const": LOCATE FN K(K)-2*(K>4),4: PRINT CHR$
(32-15*(K>4))+CHR$(90+K-N): NEXT
1090 FOR J=1 TO N: FOR K=1 TO N+1
1100 LOCATE 1,22
1110 PRINT "Introduzca los valores ";CHR$(96+K+(K=N+1)*(K+23))+FN A$(J)
+"=";SPC(9);:LOCATE 28,22: INPUT E[J,K]
1120 LOCATE (1-FN K(K))*(K<>N+1)-(K=N+1)*32,2*J+4+(K>4)*(N>4):
PRINT USING "&"; FN BS$(7)+STR$(E[J,K])
1130 NEXT K,J
1140 LOCATE 1,22: PRINT SPACE$(40)
1150 LOCATE 1,22: PRINT "  OK?   (S/N)";: INPUT K$
1160 K$=UPPER$(K$): IF K$="S" THEN RETURN ELSE 1090
```

Ejemplo

```
    N=5
                    Sus ecuaciones son

V/Z     W       X       Y       Const
a1*v + b1*w + c1*x + d1*y +
e1*z                            =k1
a2*v + b2*w + c2*x + d2*y +
e2*z                            =k2
```

```
a3*v  + b3*w  + c3*x  + d3*y  +
e3*z                            =k3
a4*v  + b4*w  + c4*x  + d4*y  +
e4*z                            =k4
a5*v  + b5*w  + c5*x  + d5*y  +
e5*z                            =k5

Introduzca los valores

V/Z     W       X       Y       Const
15    + 25    + 9.5   ± 5     +
7                               =27
9.6   ± 10    + 4.3   + 7     +
2                               =51
3.7   + 16    + 6.9   + 13    +
4                               =97
16    + 91    + 2     + 81    +
6                               =115
5     + 2     + 15    + 6     +
8                               =23

OK? (S/N)
```

18. Entrada de datos (matrices)

Esta subrutina permite introducir y comprobar una matriz de números. La presentación inicial se realiza en formato algebraico. FN A$(A) se usa para darle a la impresión un formato empaquetado. En la línea 1030 se insertan los valores numéricos empleando la función BS$(A) (borrado y retroceso). Si se comete un error, se puede cambiar los valores volviendo a la línea 1030.

```
10 REM Entrada de datos (matrices)
20 DEFINT J,K,M,N
30 DEF FN A$(A)=MID$(STR$(A),2)
40 DEF FN BS$(A)=SPACE$(A)+STRING$(A,8)
50 CLS: PRINT: PRINT " Introduzca el tamaño de la matriz como m filas y n
columnas"
60 LOCATE 1,5: INPUT "m=";M: LOCATE 20,5: INPUT "n=";N: IF N=0 OR M=0
THEN 50
70 CLS: PRINT: PRINT "     La matriz es"
80 DIM A[M,N]: K$="Y"
85 GOSUB 1000: GOSUB 2000: END
1000 FOR J=1 TO M: FOR K=1 TO N
1010 LOCATE 5*K-3,2*J+3: PRINT "a"+FN A$(J)+FN A$(K)
1020 NEXT K,J
```

```
1030 FOR J=1 TO M: FOR K=1 TO N
1040 LOCATE 1,20: PRINT "     Ahora escriba los valores numericos";
1050 IF K$<>"Y" THEN PRINT     " correctos"
1060 PRINT: PRINT "a"+FN A$(J)+FN A$(K)+" es ";FN BS$(8);: INPUT A[J,K]
1070 LOCATE 5*K-3,2*J+3: PRINT FN BS$(5);A[J,K]
1080 NEXT K,J
1090 LOCATE 1,20: PRINT FN BS$(80);"     OK?     (S/N)";: INPUT K$: K$
=UPPER$(K$)
1100 IF K$<>"S" THEN 1030
1110 RETURN
2000 CLS: TAG: MOVE 124,395: PRINT "L A     M A T R I Z     E S";
2010 PLOT 124,378: DRAWR 368,0
2020 FOR J=1 TO M: FOR K=1 TO N
2030 MOVE 80*K-40,400-40*J: PRINT A[J,K];: NEXT K,J
2040 MOVE 72,368: DRAWR -40,0: DRAWR 0,-32-40*(M-1): DRAWR 40,0
2050 MOVE 80*N+40,368: DRAWR 40,0: DRAWR 0,-32-40*(M-1): DRAWR
-40,0
2060 TAGOFF: RETURN
```

Ejemplo

Introduzca el tamaño de la matriz como m filas y n columnas

m=? 5   n=? 4

```
                a11  a12  a13  a14
                a21  a22  a23  a24
La matriz es:   a31  a32  a33  a34
                a41  a42  a43  a44
                a51  a52  a53  a54
```

Ahora escriba los valores numéricos

a11 es? 1

```
 1  3  5   7
 2  4  6   8
-1  4  6   2.5
 3  9  11  17
 6  5  3   1
```

0K? (S/N) S

$$\text{La matriz es:}\quad \begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \\ -1 & 4 & 6 & 2.5 \\ 3 & 9 & 11 & 17 \\ 6 & 5 & 3 & 1 \end{bmatrix}$$

19. Entrada de datos (variable sencilla)

Esta subrutina se emplea para introducir datos que serán empleados en posteriores análisis estadísticos.

Trabajando con variables sencillas, usted tendrá a menudo demasiados datos para visualizar al mismo tiempo. En esta rutina los datos se almacenan en V[N]. Sin embargo, antes de almacenarlos las características numéricas de cada uno se comprueban en SUB 1120. El uso de INKEY$ permite incorporar el número carácter a carácter antes de la comprobación, evitándose así las complicaciones derivadas de INPUT A$ y de una coma.

Una vez comprobados todos los números, éstos se presentan en una columna, usando VPOS(#0) para detener el flujo de datos y permitir la comprobación de 20 números al mismo tiempo.

Si hay que introducir algún cambio (línea 1250), se comprueba el número de subíndice de la variable (línea 1330) antes de pedir el nuevo valor (líneas 1340-1360).

Los veinte valores revisados se vuelven a mostrar por si fuera necesaria una corrección posterior. Si no es así, la rutina muestra los veinte valores siguientes, y así sucesivamente.

```
10 REM Introduccion de datos (variable sencilla)
20 DEFINT A,K,N,P,Q,S,T
30 DEF FN A$(A)=MID$(STR$(A),2)
40 A$=STRING$(10,32)+STRING$(10,8): NL$=CHR$(10)+CHR$(13)
50 CLS: PRINT: INPUT "Cuantas variables hay";N
60 IF N>=4 THEN 65 ELSE PRINT STR$(N)+" no hay valores suficientes para
llevar a cabo un analisis estadistico adecuado.": PRINT NL$;"Para comenzar de
nuevo pulse cualquier tecla.": GOSUB 1100: CLS: GOTO 50
65 GOSUB 1000: GOSUB 2000: END
1000 DIM V[N]: CLS
1010 WINDOW #2,1,40,22,25
1020 PRINT #2, "Introduzca los valores"
1030 FOR P=1 TO N: V$=""
1040 LOCATE #2,1,2: PRINT #2,"  V(";FN A$(P);") es ";: PRINT #2,A$;
1050 T$=INKEY$: IF T$="" THEN 1050
1060 PRINT #2,T$;: IF T$=CHR$(13) THEN 1070 ELSE V$=V$+T$: GOTO 1050
1070 GOSUB 1120: IF FAIL THEN P=P-1 ELSE V[P]=VAL(V$)
1080 NEXT
1090 CLS :GOSUB 1170: RETURN
1100 IF INKEY$="" THEN 1100
1110 RETURN
1120 FAIL=0: S=0: T=1: IF LEFT$(V$,1)="+" OR LEFT$(V$,1)="-" THEN T=2
1130 FOR Q=T TO LEN(V$): Z$=MID$(V$,Q,1)
1140 IF Z$="." THEN S=S+1
```

```
1150 IF Z$="." OR Z$>="0" AND Z$<="9" THEN NEXT: IF S<=1 THEN
RETURN
1160 FAIL=-1: RETURN
1170 A=1
1180 PRINT: PRINT #2, "Son correctos estos valores?": FOR P=A TO N
1190 PRINT TAB(10);"V(";FN A$(P);")=";V[P]
1200 IF VPOS(#0)>20 THEN 1220
1210 NEXT
1220 GOTO 1240
1225 IF P>N THEN RETURN
1230 CLS: GOTO 1190
1240 PRINT #2, "(S/N)";
1250 INPUT #2,K$: K$=UPPER$(K$)
1260 IF K$="S" THEN A=P+1: GOTO 1225
1270 CLS #2: PRINT #2,"Que entrada desea cambiar?";
1280 V$=""
1290 T$=INKEY$: IF T$="" THEN 1290
1300 PRINT #2,T$;: IF T$=CHR$(13) THEN 1310 ELSE V$=V$+T$: GOTO 1290
1310 GOSUB 1120: IF FAIL THEN 1270
1320 IF LEFT$(V$,1)<>"-" THEN K=VAL(V$) ELSE 1270
1330 IF K>N OR K>A+18 OR K<A OR K<=0 THEN PRINT "No es valido":
GOTO 1270
1340 CLS #2: V$="": PRINT #2,"Introduzca el valor correcto V, (";FN
A$(ABS(K));")=";
1350 T$=INKEY$: IF T$="" THEN 1350
1360 PRINT #2,T$;: IF T$=CHR$(13) THEN 1370 ELSE V$=V$+T$: GOTO 1350
1370 GOSUB 1120: IF FAIL THEN PRINT "Entrada incorrecta": GOTO 1340
1380 V[K]=VAL(V$): CLS: GOTO 1180
2000 CLS: PRINT: FOR P=1 TO N
2010 PRINT TAB(10);"V[";FN A$(P);"]=";V[P]: NEXT
2020 RETURN
```

20. Entrada de datos (coordenadas x e y, datos estadísticos, etc.)

Esta subrutina acepta pares de valores, permite su comprobación y corrección y, por último, los almacena.

El problema de tener demasiados datos puede subsanarse si se los maneja en grupos de diez. La matriz B se elige para que sea un múltiplo de diez, y puede ser tan grande como sea necesario para albergar los datos. Los números se introducen usando INKEY$ para cada dígito, y las características numéricas de cada número, una vez completo éste, se comprueban en SUB 1400.

Cuando se ha completado la operación de introducción, los números son presentados en grupos de diez, y se los puede modificar si es necesario. Después de haber

hecho la corrección, se puede presentar cada grupo hasta que usted esté satisfecho. R se usa como contador de bloque.

Una vez que está usted satisfecho con los datos, éstos son transferidos a la matriz E, que tiene el tamaño correcto.

```
10 REM Introduccion de datos (coordenadas x e y)
20 DEFINT L,N,P,Q,R,S: DEF FN A$(A)=MID$(STR$(A),2)
30 BS$=STRING$(10,32)+STRING$(10,8)
40 CLS: LOCATE 2,2: INPUT "Numero de pares que leer";N: IF N<=0 THEN 40
50 DIM E[1,N]: GOSUB 1000: END
1000 DIM B[1,10*INT((N-1)/10+1)]:CLS
1010 WINDOW #1,1,40,18,21: WINDOW #2,1,40,22,25
1020 LOCATE #1,1,1: PRINT #1,"  Escriba los valores de las coordenadas x e y"
1030 FOR P=1 TO N: P$=FN A$(P)
1040 LOCATE #2,5,1: PRINT #2,BS$;"x";P$;"=";: Z$=""
1050 T$=INKEY$: IF T$="" THEN 1050
1060 PRINT #2,T$;
1070 IF T$<>CHR$(13) THEN Z$=Z$+T$:GOTO 1050
1080 GOSUB 1400: IF FAIL THEN 1040 ELSE B[0,P]=VAL(Z$)
1090 LOCATE #2,5,3: PRINT #2,BS$;"y";P$;"=";: Z$=""
1100 T$=INKEY$: IF T$="" THEN 1100
1110 PRINT #2,T$;
1120 IF T$<>CHR$(13) THEN Z$=Z$+T$:GOTO 1100
1130 GOSUB 1400: IF FAIL THEN 1090 ELSE B[1,P]=VAL(Z$)
1140 CLS #2: NEXT
1150 R=0: CLS
1160 FOR P=R TO INT((N-1)/10): R=P
1170 FOR Q=1 TO 10: QP=Q+10*P: Q$=FN A$(QP)
1180 LOCATE 4,Q+3: PRINT BS$;"x";Q$;"=";B[0,QP]
1190 LOCATE 20,Q+3: PRINT BS$;"y";Q$;"=";B[1,QP]
1200 NEXT
1210 LOCATE #1;2,1: INPUT #1,"Son correctos estos datos (s/n)";K$: CLS #1
1220 K$=UPPER$(K$): IF K$="S" THEN 1360
1230 LOCATE #1,2,1: PRINT #1,"En que linea entre";1+10*p;"y";10*(P+1);
"quiere cambiar algo";: INPUT #2,L: CLS #1: CLS #2
1240 IF L>=1+10*P AND L<=10*(P+1) THEN 1260 ELSE LOCATE #1,2,1:
PRINT #1,"Linea incorrecta, pruebe otra vez"
1250 INPUT #2,L: CLS #1: CLS #2: GOTO 1240
1260 LOCATE #1,2,1: PRINT #1,"Introduzca los valores correctos"
1270 LOCATE #2,5,1: PRINT #2,"x";FN A$(L);"=";: Z$=""
1280 T$=INKEY$: IF T$="" THEN 1280
1290 PRINT #2,T$;: IF T$<>CHR$(13) THEN Z$=Z$+T$:GOTO 1280
1300 GOSUB 1400: IF FAIL THEN CLS #2: GOTO 1270 ELSE B[0,L]=VAL(Z$)
1310 LOCATE #2,5,3: PRINT #2,"y";FN A$(L);"=";: Z$=""
1320 T$=INKEY$: IF T$="" THEN 1320
```

```
1330 PRINT #2,T$;: IF T$<>CHR$(13) THEN Z$=Z$+T$: GOTO 1320
1340 GOSUB 1400: IF FAIL THEN CLS #2: GOTO 1310 ELSE B[1,L]=VAL(Z$)
1350 CLS #1: CLS #2: GOTO 1160
1360 NEXT
1370 FOR P=1 TO N
1380 E[0,P]=B[0,P]: E[1,P]=B[1,P]: NEXT
1390 ERASE B: RETURN
1400 FAIL=0: S=0: T=1: IF LEFT$(Z$,1)="+" OR LEFT$(Z$,1)="-" THEN T=2
1410 IF Z$="+" OR Z$="-" OR Z$="." THEN 1460
1420 FOR A=T TO LEN(Z$): X$=MID$(Z$,A,1)
1430 IF X$="." THEN S=S+1
1440 IF X$<>" " AND X$<>"." AND (X$<"0" OR X$>"9") THEN 1460
1450 NEXT: IF S<=1 THEN RETURN
1460 FAIL=-1: RETURN
```

21. Fichero de imagen

La relación entre el mapa de memoria y la presentación en la pantalla es bastante complicada y hace que las funciones de PEEK (examinar el contenido de una posición de memoria) y POKE (modificar el contenido de una posición) deban ser empleadas con mucho cuidado. El sencillo programa siguiente muestra bastantes cosas sobre la forma en que funciona la imagen, y también sobre sus complicaciones.

```
10 MODE 1
20 Z=&C000
30 FOR P=0 TO 255
40 POKE Z,P: Z=Z+1
50 NEXT: IF Z<-128 THEN 30
60 END
```

En principio, el origen inicial de la pantalla, Z, está en el extremo superior izquierdo. Los bytes sucesivos llenan la línea superior de cada posición de impresión de la pantalla. Hay 80 bytes que atraviesan la pantalla en cada modo, lo que, con 25 líneas, supone un total de 25×80=2000 bytes. La memoria está dividida en 8 bloques, cada uno de una longitud de 2048 bytes, de forma que los últimos 48 bytes de cada bloque no se utilizan. Un bloque nuevo llena la segunda línea de la posición de impresión, y así sucesivamente. El diagrama aclara esta explicación.

En modo 1 (MODE 1) hay cuatro colores, pero éstos quedan reducidos a dos en modo 2 (MODE 2). En modo 0 (MODE 0; cambie la línea 10) hay 16 colores con algún parpadeo. Los colores se controlan por referencia a cuáles bits son 0 y cuáles son 1.

El diagrama muestra qué bits están implicados con qué pixels en los tres modos. En modo 2 un pixel se asocia con un bit de forma que sólo son posibles dos colores, ya que un bit puede estar desactivado (tinta 0) o activado (tinta 1) para dar los colores del fondo y del primer plano. En modo 1 los pixels son dos veces mayores y están

controlados por dos bits; de esta forma son posibles cuatro colores. La tabla siguiente muestra las relaciones.

Tinta 0 (INK 0) — ambos bits desactivados
Tinta 1 (INK 1) — bit superior activado, bit inferior desactivado
Tinta 2 (INK 2) — bit superior desactivado, bit inferior activado
Tinta 3 (INK 3) — ambos bits activados

Estas relaciones corresponden a la representación binaria de 0-3. Una combinación similar se aplica al modo 0, en el que son posibles 16 colores correspondientes a las formas binarias de 0-15.

Estimulando los tres fósforos, rojo, azul y verde, con diferentes intensidades se consigue una paleta de 27 colores.

La segunda dificultad con la imagen de la pantalla es que ésta se mueve desplazando su origen relativo a &C000, el origen inicial. El siguiente programa muestra una sentencia POKE directa para producir un subrayado.

```
10 CLS
20 LOCATE 10,10
30 PRINT "QWERTY"
40 FOR N=0 TO 11
50 POKE N+&C000+818,240
60 NEXT: PRINT
70 STOP
```

Sin embargo, el desplazamiento de la pantalla y la reejecución separan el subrayado de la palabra, y es necesario volver a establecer el origen de la pantalla en &C000 con las rutinas de código de máquina SCR SET BASE y SCR SET OFFSET. Para esto se puede emplear las líneas siguientes:

```
1000 RESTORE 1000: DATA &E5,&F5,&E3,$C0,&CD,&08,&BC,&21,&00,&00,
&CD,&05,&BC,&E1,&F1,&C9
1010 MEMORY 40000
1020 A=40200
1030 FOR N=0 TO 15
1040 READ Z
1050 POKE A+N,Z
1060 NEXT
```

Ejecute primero el programa de código de máquina (con RUN 1000) para establecer la subrutina; a continuación dé la orden RUN 10. Después del desplazamiento, vuelva a dar la orden RUN 10. Escriba CALL 402000 para restablecer las condiciones originales. Estas técnicas pueden ser útiles para conseguir efectos especiales, pero no para el uso corriente.

Otra característica de la memoria RAM es que los gráficos del usuario están localizados por encima de HIMEM, de forma que es posible modificar posiciones de RAM directamente con POKE sin tener que usar SYMBOL. Esto queda ilustrado en el apartado 'Impresión de doble tamaño'.

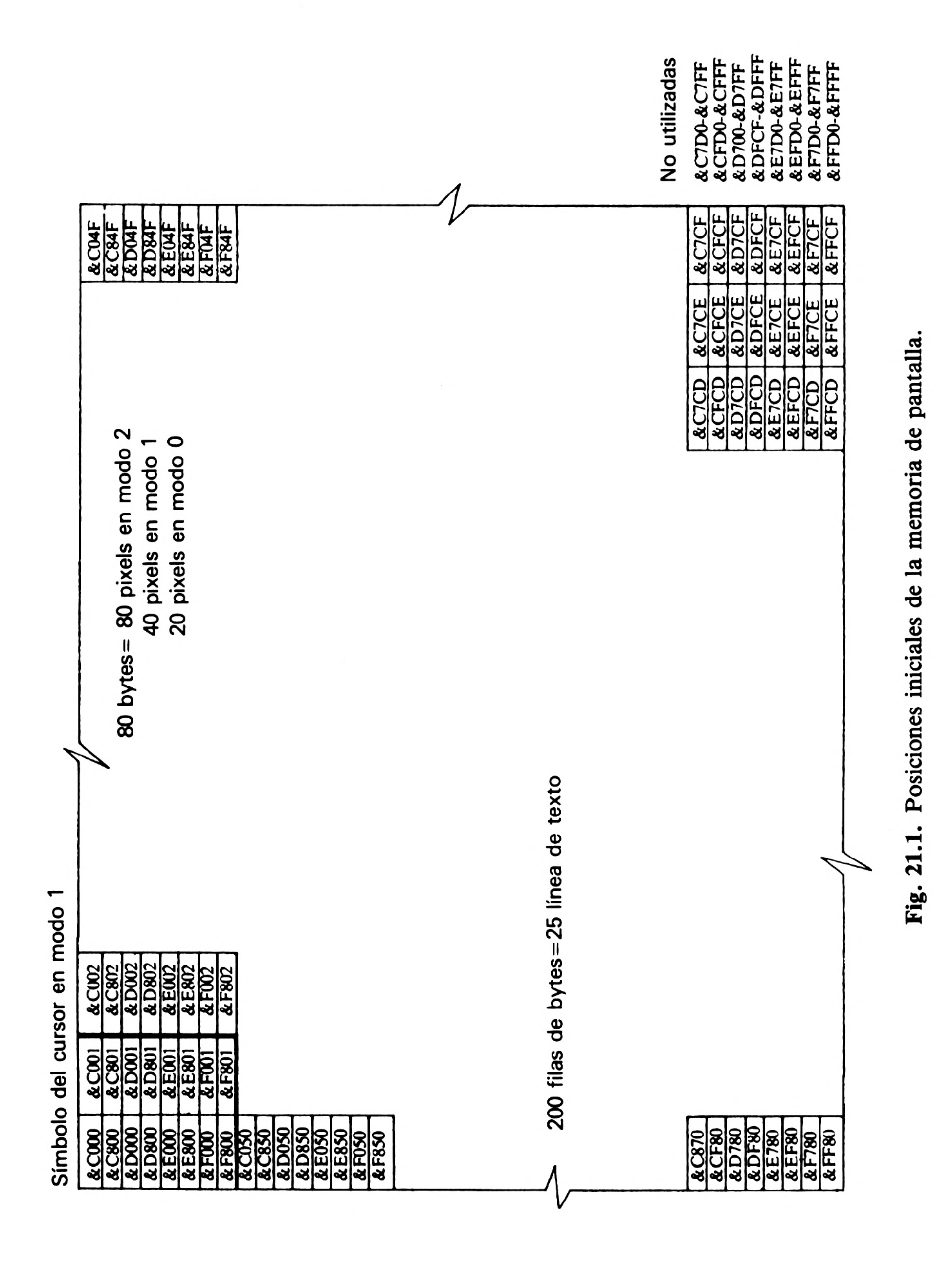
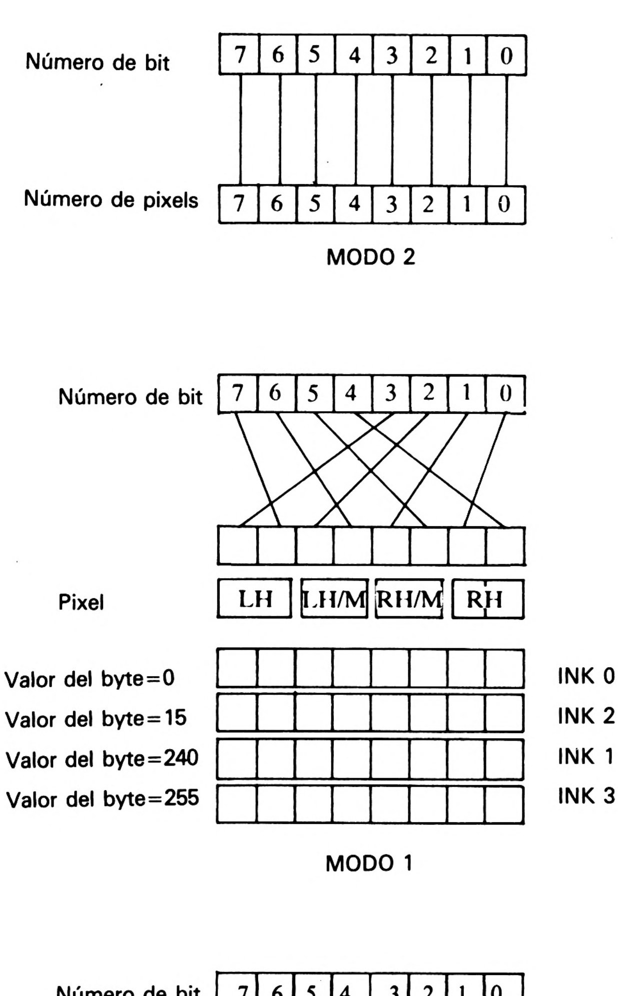


Fig. 21.1. Posiciones iniciales de la memoria de pantalla.



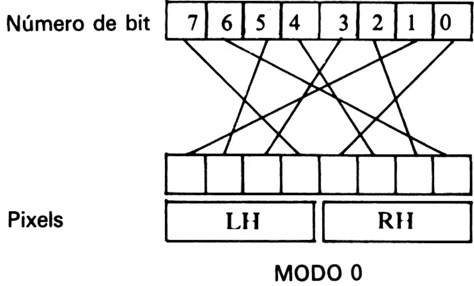


Fig. 21.2. Relaciones entre bits y colores en cada momento.

22. Impresión de doble tamaño

SYMBOL AFTER 64 coloca la información ASCII sobre las mayúsculas inmediatamente después de HIMEM. El programa funciona examinando (PEEK) grupos sucesivos de ocho bytes, que representan las letras mayúsculas, en la posición ADDR. Las matemáticas convierten esta información en modelos de 32 bytes, y éstos se almacenan en grupos de cuatro en la segunda mitad del conjunto ASCII, que contiene los modelos geométricos y las letras extranjeras.

Cuando se pide que una letra mayúscula tenga doble tamaño, el programa comprueba primero que se trata de una letra y que hay espacio para imprimirla. Si es así, imprime cuatro bytes que representan los cuartos de la letra, dos de ellos en dos posiciones adyacentes de una línea y los otros dos en la línea de abajo. Los espacios se separan e imprimen con CHR$(32).

Lleva alrededor de 40 segundos establecer la información inicial de los gráficos definidos por el usuario.

```
10 REM Impresion de doble tamaño (solamente MAYUSCULAS y espacios)
20 GOSUB 1000: GOSUB 2000: END
1000 SYMBOL AFTER 64: H=HIMEM+1: H1=H+512
1010 DIM T[1,16]
1020 FOR P=1 TO 16: T[0,P]=P-1: READ T[1,P]: NEXT
1030 RESTORE 1030: DATA 0,3,12,15,48,51,60,63,192,195,204,207,240,243,252,
255
1040 FOR A=1 TO 26: ADDR=H+8*A: B=A*32+H1
1050 FOR P=0 TO 7: T=PEEK(ADDR+P): I=INT(T/16): J=T-16*I
1060 FOR Z=1 TO 16
1070 IF T[0,Z]=J THEN T1=T[1,Z]
1080 IF T[0,Z]=I THEN T2=T[1,Z]
1090 NEXT
1100 C=B+2*P
1110 POKE C,T2: POKE C+1,T2: POKE C+16,T1: POKE C+17,T1
1120 NEXT P,A
1130 RETURN
2000 CLS: PRINT "Introduzca lo que quiera imprimir": PRINT: INPUT A$
2010 A$=UPPER$(A$)
2020 FOR P=1 TO LEN(A$): Z=ASC(MID$(A$,P,1))
2030 IF (Z<65 OR Z>90) AND Z<>32 THEN PRINT " No son mayusculas. Pulse
una tecla para volver a introducir el texto.": GOSUB 2170: GOTO 2000
2040 NEXT
2050 PRINT "Introduzca las posiciones en las que hay que escribir el texto"
2060 PRINT: INPUT "Numero de fila (1-23)  ";R: IF R<1 OR R>23 THEN 2060
2070 PRINT: INPUT "Numero de columna (1-39)";C: IF C<1 OR C>39 THEN 2070
2080 R=2*INT(R/2)+1: C=2*INT(C/2)+1
```

```
2090 IF LEN(A$)>240-(C-1)/2-10*(R-1) THEN PRINT " El texto es demasiado
largo para ser impreso en el area de pantalla. Pulse una tecla para volver a
introducirlo.":GOSUB 2170: GOTO 2000
2100 X=C: Y=R: CLS
2110 FOR P=1 TO LEN(A$): Z=(ASC(MID$(A$,P,1))-64)*4
2120 LOCATE X,Y: IF Z<>-128 THEN PRINT CHR$(Z+128);CHR$(Z+130) ELSE
PRINT CHR$(32)
2130 LOCATE X,Y+1: IF Z<>-128 THEN PRINT CHR$(Z+129);CHR$(Z+131)
ELSE PRINT CHR$(32)
2140 X=X+2: IF X>39 THEN X=1: Y=Y+2
2150 NEXT
2160 RETURN
2170 IF INKEY$="" THEN 2170
2180 RETURN
```

23. Trazado de líneas entre dos puntos

El dibujo de líneas en Amstrad BASIC es muy sencillo si se usan las órdenes PLOT y DRAW (o PLOTR y DRAWR). Dado que las coordenadas están situadas dentro de los 16 bits de capacidad (65535), el sistema dibuja automáticamente la línea, que aparecerá en la pantalla si el área de ésta ($0 \geqslant x \geqslant 639$, $0 \geqslant y \geqslant 399$) contiene a la línea total o parcialmente.

Sin embargo, es importante recordar que la relación de aspecto de la pantalla es 0.92, de forma que, si usted desea dibujar cualquier figura con su perfil verdadero, debe tener en cuenta esto en las coordenadas. Por ejemplo,

```
PLOT 150,40
DRAWR 350,0: DRAWR 0,322
DRAWR -350,-0: DRAWR 0,-322
```

dibuja en la pantalla un verdadero cuadrado a pesar de que el número de pixels es diferente. En efecto, en la figura son ligeramente rectangulares. Para dibujar una elipse, la ecuación debe tener en cuenta tanto la relación de aspecto de la circunferencia que da la elipse como la de la pantalla.

El dibujo de una línea de puntos es un poco más difícil, ya que surgen dos problemas. En primer lugar, trabajar con líneas tanto verticales como horizontales con una sola rutina provocará la división por cero con una de ellas. Así, son necesarias dos rutinas. El segundo problema es guardar la misma relación trazo/espacio para líneas en ángulos diferentes.

La línea 2070 proporciona la línea horizontal y la 2030 la vertical. P es la pendiente, aunque si es menor que uno se usa el recíproco. I es el intervalo entre cada nuevo comienzo del trazo, y se ha establecido un bucle en Q para PLOT X1,Y1 (dibujar X1,Y1) y después para DRAW 0.4 (dibujar 0.4 de la distancia al siguiente punto de trazo), y así sucesivamente. Una relación trazo/espacio de 0.4/0.6 parece dar mejores resultados que la de 0.5/0.5.

SGN en I cuida de la dirección del trazo, es decir, L hacia R o R hacia L, y SQR(1+P*P) mantiene constante la relación trazo/espacio para diferentes pendientes de línea.

Esta subrutina dibuja una línea de puntos entre (X1,Y1) y (X2,Y2).

```
10 REM Linea discontinua
20 INPUT "X1=";X1: INPUT "Y1=";Y1: INPUT "X2=";X2: INPUT "Y2=";Y2
30 CLS: GOSUB 2000: END
2000 IF X1=X2 THEN 2070 ELSE IF Y1=Y2 THEN 2030
2010 P=(Y2-Y1)/(X2-X1)
2020 IF ABS(P)<1 THEN 2040 ELSE P=1/P: GOTO 2080
2030 P=0
2040 I=10*SGN(X2-X1)/SQR(1+P*P)
2050 FOR Q=0 TO (X2-X1)/I
2060 PLOT X1+Q*I,Y1+Q*I*P: DRAWR 0.4*I,0.4*I*P: NEXT: RETURN
2070 IF Y1=Y2 THEN RETURN ELSE P=0
2080 I=10*SGN(Y2-Y1)/SQR(1+P*P)
2090 FOR Q=0 TO (Y2-Y1)/I
2100 PLOT X1+Q*I*P,Y1+Q*I: DRAWR 0.4*I*P,0.4*I: NEXT: RETURN
```

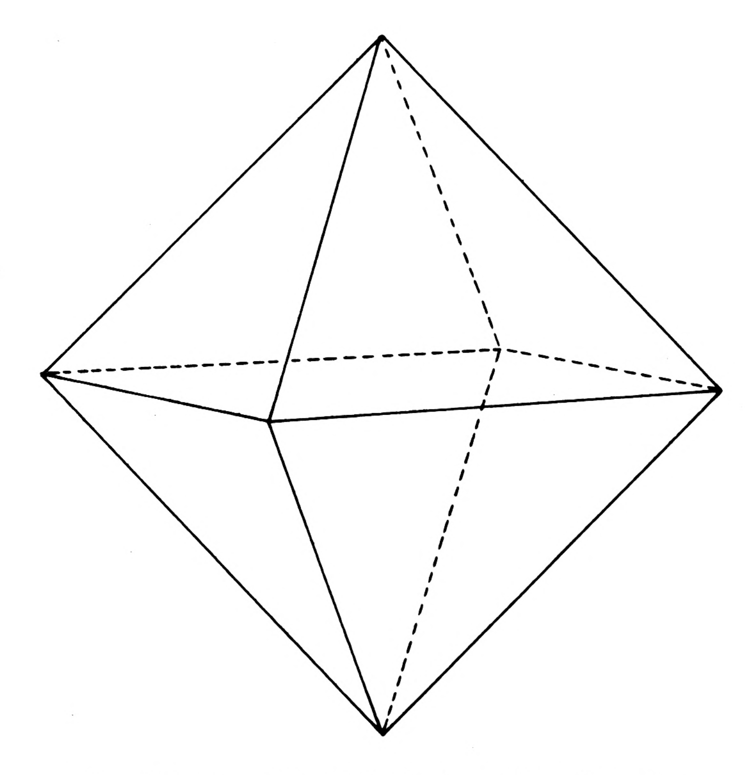


Fig. 23.1. Ilustración del dibujo de una línea discontinua.

24-26. Errores

Si usted quiere simular datos que sigan una ley de error conocida, sería conveniente poder generar números relacionados con distribuciones particulares. Por ejemplo, para circunstancias en las que se combinan bastantes errores aleatorios pequeños, como al hacer una medida de longitud, la distribución de Gauss parece ser una de las más próximas a los resultados observados. Si la probabilidad de un suceso es muy pequeña, por ejemplo el número de accidentes de carretera fatales que ocurren en un periodo dado (comparado con el número total de viajes que se hacen en ese periodo), entonces es mucho más relevante la distribución de Poisson. Las selecciones referentes a lanzamientos de dados o manos de cartas siguen una distribución binomial.

La facilidad del generador de números aleatorios es una distribución rectangular, de forma que cada número del intervalo seleccionado tiene la misma probabilidad de ocurrir. Estas subrutinas alteran esta distribución, y así es más probable que los números estén más cerca de la media que de los extremos de la distribución apropiada.

El gráfico muestra los perfiles típicos de las distribuciones.

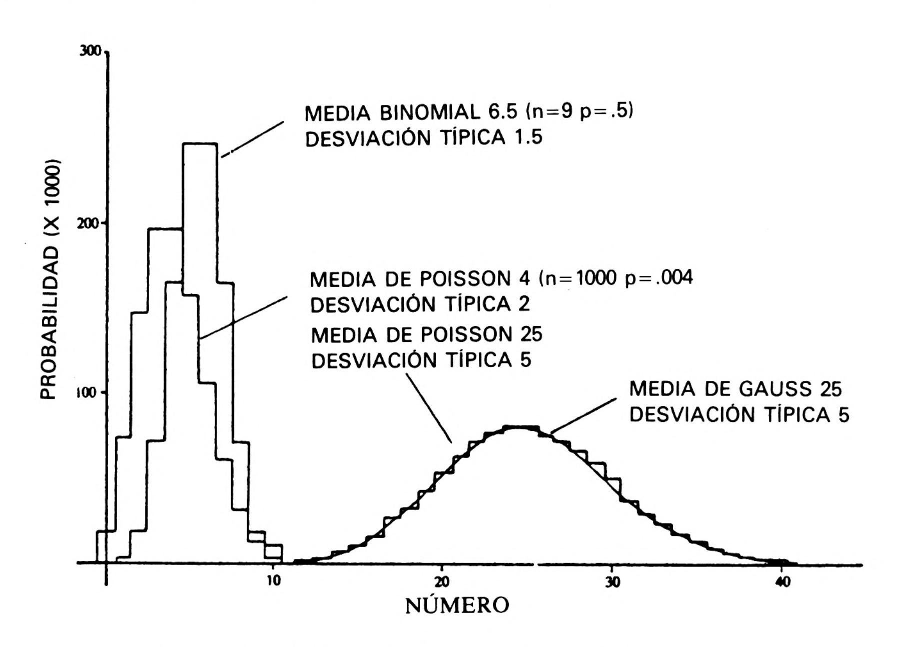


Fig. 24/26.1. Comparación de distribuciones diferentes.

24. Distribución binomial

Esta rutina crea una matriz G que, probada aleatoriamente, da un conjunto de valores distribuidos binomialmente. Como el tamaño de la matriz está limitado por la capacidad de memoria, la rutina sólo funciona con valores pequeños de N, como indica la expresión para Y. La distribución binomial va aproximándose a la distribución de Gauss al ir aumentando N.

```
10 REM Distribucion binomial
20 INPUT "Escriba la probabilidad de que ocurra el suceso, que debe estar en el margen de 0 a 1";P
30 IF P<=0 OR P>=1 THEN 20
40 INPUT "Escriba el numero de elementos en la seleccion";N
50 IF N<>ABS(INT(N)) OR N=0 THEN 50
60 GOSUB 1000: GOSUB 2000: END
1000 DIM B[N+3]: T=P/P-1
1010 IF P>0.5 THEN T=1/T
1020 Z=1: M=N
1030 FOR A=1 TO N+1: B[A]=INT(Z+0.5)
1040 Z=Z*M*T/A: M=M-1: NEXT A
1050 Y=INT(1/P↑N): IF P>0.5 THEN Y=INT(1/(1-P)↑N)
1060 DIM G[Y]: A=1: B=0: Q=B[A]
1070 FOR C=1 TO Y: G[C]=B
1080 IF C>=Q THEN A=A+1: B=B+1: Q=Q+B[A]
1090 NEXT C: IF P<0.5 THEN FOR C=1 TO Y: G[C]=N-G[C]: NEXT C
1100 RETURN
2000 PRINT G[INT(RND(1)*Y)+1];: IF INKEY$="" THEN 2000
2010 RETURN
```

25. Distribución de Gauss

Esta rutina establece una matriz de 1823 posiciones de memoria que, probadas aleatoriamente, generarán valores distribuidos en una curva normal en torno a una media elegida y con una desviación típica dada.

```
10 REM Distribucion de Gauss
20 DEF FN G(X)=90.0013*EXP(-X*X/2)
30 DIM G[1823]: GOSUB 1000
40 INPUT "Escriba el valor deseado de la media";M
50 INPUT "Escriba la desviacion tipica deseada";S
60 IF S<=0 THEN 50
70 GOSUB 2000: END
1000 N=1
1010 FOR P=-3 TO 3 STEP 0.125
1020 E=(FN G(P)+FN G(P+0.125))/2
```

```
1030 FOR Q=0 TO E
1040 G[N]=P+RND(1)*0.125: N=N+1
1050 NEXT Q,P
1060 RETURN
2000 Z=M+S*G[INT(RND(1)*1823)+1]
2010 PRINT ROUND(Z,4): IF INKEY$="" THEN 2000
2020 RETURN
```

Ejemplo

```
run
Escriba el valor de la media deseado?
Escriba la desviación deseada?

35.0681
34.0545
34.6708
35.0419
31.6583
33.8876
34.7117
33.6309
33.2394
35.9174
32.6720
36.9178
32.3196
34.8433
36.2537
35.0322
35.5951
*Break*
```

26. Distribución de Poisson

En la distribución de Poisson, la probabilidad de que ocurran exactamente N sucesos en un periodo dado se obtiene mediante

P=M↑N/N!*EXP(−M)

donde M es el promedio de sucesos en el periodo.

Ésta es la fórmula empleada en la subrutina. La desviación típica es igual a SQR(M) y la distribución se lleva tan lejos como 3*SQR(M), es decir, N3.

```
10 REM Distribucion de Poisson
20 INPUT "Escriba el numero medio de veces que ocurre un suceso en un periodo
dado";M: IF M>86 THEN 20
30 GOSUB 1000: CLS :GOSUB 2000: END
1000 M=ABS(M): N3=3*SQR(M)
1010 N=2*N3+(N3-M)*(M<N3)
1020 DIM A[N+1,2]: K=1/M: T=1
1030 FOR P=0 TO M+N3
1040 K=K*M/(P-(P=0))
1050 Z=K*EXP(-M)
1060 IF P>M-N3 THEN A[T,1]=Z: A[T,2]=P: T=T+1
1070 NEXT
1080 DIM B[N+2]: Z=0
1090 FOR P=1 TO N+1
1100 B[P]=INT(A[P,1]*1000+0.5)
1110 Z=Z+B[P]: NEXT
1120 DIM P[Z]: T=1: S=B[T]
1130 FOR P=1 TO Z: P[P]=A[T,2]
1140 IF P>=S THEN T=T+1: S=S+B[T]
1150 NEXT
1160 RETURN
2000 PRINT "El numero de veces que ocurre un suceso en subsiguientes periodos
viene dado por:"
2010 PRINT P[INT(RND(1)*Z+1)];: IF INKEY$="" THEN 2010
2020 ERASE A,B,P: RETURN
```

27. Evaluación de un determinante

Esta elegante subrutina se basa en el hecho de que en el método de Gauss-Jordan de inversión de matrices (véase 'Inversión de matrices') la entrada inferior derecha de la matriz, al final del ciclo 'siguiente-al-último' del bucle externo, es el determinante que usted quiere dividido por el número menor de esta entrada (es decir, el determinante original con la columna RH y la fila inferior omitidas). Usted no conoce el valor de este número menor, pero ya ha sido calculado de la misma forma al final del ciclo 'segundo-al-último' del bucle externo, dando el menor número del número menor como la incógnita. La repetición de este procerso N – 1 veces tiene finalmente como resultado la entrada superior LH que usted conoce. Por tanto, la respuesta es el producto de cada uno de estos pasos.

DEF FN sirve para imprimir las entradas de determinantes A[1,1], etc. en formato empaquetado.

La primera línea de la subrutina proporciona un determinante con un solo término (orden 1).

SUB 1140 y SUB 1180 condicionan al determinante para evitar un fallo innecesario en la línea 1050, donde aparecerá el mensaje 'Division by zero' ('división por cero') si A[I,I]=0 y el IF de protección no estuviese presente en la línea 1050.

SUB 1140 se asegura de que las entradas diagonales no son ceros, y SUB 1180 añade la fila inferior a la fila superior y la columna RH a la columna LH si el determinante superior LH 2×2 es cero (esto no altera el valor del determinante).

Las líneas 1030 a la 1120 invierten la matriz y multiplican los productos de cada nivel por el ciclo $N-1$ de la variable I.

Hay una resta en la línea 10 cuyo resultado puede ser un cero, y esto provocaría un fallo en el siguiente ciclo a menos que fuese detectado en la línea 1110. Ésta abandona el programa, ya que el valor del determinante será cero, lo que ocurre, por ejemplo, si dos filas o dos columnas son iguales o si una es múltiplo de la otra (véase el segundo ejemplo).

La subrutina obtiene el valor numérico de un determinante $|A|$ de orden N.

```
10 REM Evaluacion de un determinante
20 DEFINT I,J,K,N,P,T:DEF FN A$(A)=MID$(STR$(A),2)
30 INPUT "Introduzca el orden del determinante";N: IF N<1 THEN 30
40 DIM A[N,N]: FOR I=1 TO N: FOR J=1 TO N
50 PRINT "A["+FN A$(I)+","+FN A$(J)+"]=";: INPUT A[I,J]: NEXT J,I
60 GOSUB 1000: GOSUB 2000: END
1000 D=A[1,1]: IF N=1 THEN RETURN
1010 GOSUB 1140
1020 IF A[1,1]*A[2,2]=A[1,2]*A[2,1] THEN GOSUB 1180
1030 D=A[1,1]*T: FOR P=N TO 2 STEP -1: DIM D[P,P]
1040 FOR I=1 TO P: FOR J=1 TO P: D[I,J]=A[I,J]: NEXT J,I
1050 FOR I=1 TO P-1: IF ABS(D[I,I])<0.00000001 THEN D=0: RETURN ELSE
D[I,I]=1/D[I,I]
1060 FOR J=1 TO P: IF J=I THEN 1110
1070 D[J,I]=D[J,I]*D[I,I]
1080 FOR K=1 TO P: IF K=1 THEN 1100
1090 D[J,K]=D[J,K]-D[J,I]*D[I,K]
1100 NEXT
1110 NEXT: IF ABS(D[I+1,I+1])<0.00000001 THEN D=0: RETURN
1120 NEXT: D=D*D[P,P]: ERASE D: NEXT
1130 RETURN
1140 T=1: FOR I=1 TO N: IF A[I,I]=0 THEN GOSUB 1160
1150 NEXT: RETURN
1160 FOR J=2 TO N: IF A[I,J]<>0 THEN T=-T: FOR K=1 TO N: SW=A[K,I]:
A[K,I]=A[K,J]: A[K,J]=SW: NEXT: RETURN
1170 NEXT: RETURN
1180 FOR I=1 TO N: A[1,I]=A[1,I]+A[N,I]: NEXT
1190 FOR I=1 TO N: A[I,1]=A[I,1]+A[I,N]: NEXT
1200 RETURN
2000 PRINT "Det D=";D
2010 RETURN
```

Ejemplos

N=4

$$\begin{vmatrix} -2 & 4 & 7 & 3 \\ 8 & 2 & -9 & 5 \\ -4 & 6 & 8 & 4 \\ 2 & -9 & 3 & 8 \end{vmatrix}$$

Det D=2084.01935

N=5

$$\begin{vmatrix} 15 & 7 & 6 & 9 & 3 \\ 2 & 4 & 6 & 8 & 10 \\ 31 & 6 & 11 & 17 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 81 & 9 & 23 & 1 & 3 \end{vmatrix}$$

Det D=154255.21

28. Factorial de n

Esta subrutina calcula el factorial de n (para $n<33$). Ilustra un bucle muy sencillo con multiplicaciones sucesivas. Los factoriales aparecen en permutaciones y combinaciones, así como en series.

```
10 REM Factorial de n
20 INPUT "Numero";N: IF N<>ABS(INT(N)) OR N>33 THEN 20
30 GOSUB 1000: GOSUB 2000: END
1000 Z=1: FOR P=1 TO N: Z=Z*P: NEXT
1010 RETURN
2000 PRINT "N!=";Z
2010 RETURN
```

29. Operador Heaviside

Esta subrutina ejecuta la operación de Heaviside: activar una función en $x=a$ y desactivarla en $x=b$.

Es muy útil en el estudio de problemas mecánicos y eléctricos, en los que se dan funciones escalón (por ejemplo, el hecho de encender un interruptor). A menudo se usa la función escalón para comprobar la estabilidad de un sistema o su respuesta. En el ejemplo se da la respuesta de un circuito que contiene un condensador y una resistencia en serie, y puede comprobarse que, a pesar de que el voltaje se enciende y se apaga instantáneamente, esto no es así para el voltaje del condensador. Lleva tiempo efectuar la carga y la descarga. Cambie el valor de RC para ver el efecto sobre la respuesta. Esta prueba se usa a menudo para calcular la ejecución de amplificadores y altavoces en sistemas de alta fidelidad.

El programa de control de estabilidad es otro ejemplo del uso del operador Heaviside.

```
10 REM Operador de Heaviside
20 DEF FN H(X,A,B)=-(X>=A)+(X>=B)
```

```
30 REM Ejemplo
40 A=100: B=198: RC=10
50 DEF FN H(X,A,B)=-(X>=A)+(X>=B)
60 FOR X=1 TO 639: T=(X-100*INT(X/100))/RC
70 PLOT X,150+128*EXP(-T)+128*FN H(X,A,B+2)*(1-2*EXP(-T))
80 PLOT X,128*FN H(X,A+1,B+1)
90 IF X=A-1 OR X=B+1 THEN DRAWR 0,128
100 IF X>B THEN A=A+200: B=B+200
110 NEXT
```

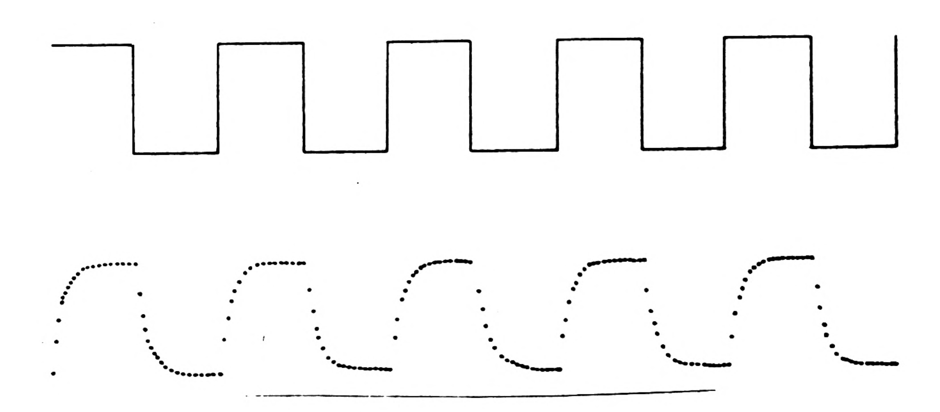


Fig. 29.1. Onda cuadrada y onda cuadrada amortiguada.



30-33. Aritmética de alta precisión

Como los cálculos sencillos llevados a cabo por los Amstrad tienen una exactitud del 90%, no parece necesario conseguir más precisión. Sin embargo, incluimos aquí este grupo de programas porque son excelentes ejemplos de las técnicas de manipulación de cadenas y cubren todas las posibilidades. Hay varias formas de aproximarse a los cálculos empleando cadenas; los adoptados aquí para la suma, resta y multiplicación siguen los métodos de cálculo sin abreviaturas. También usan tablas de consulta en las que se puede encontrar las decenas y las unidades; por ejemplo, '9'+'7'='1' decena y '6' unidades.

Las siguientes son las tablas generadas en las primeras secciones de las rutinas:

Tablas de consulta de unidades y decenas

P+Q

P \ Q	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

UNIDADES

P \ Q	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	0	1	1	1
4	0	0	0	0	0	0	1	1	1	1
5	0	0	0	0	0	1	1	1	1	1
6	0	0	0	0	1	1	1	1	1	1
7	0	0	0	1	1	1	1	1	1	1
8	0	0	1	1	1	1	1	1	1	1
9	0	1	1	1	1	1	1	1	1	1

DECENAS

P−Q

P \ Q	0	1	2	3	4	5	6	7	8	9
0	0	9	8	7	6	5	4	3	2	1
1	1	0	9	8	7	6	5	4	3	2
2	2	1	0	9	8	7	6	5	4	3
3	3	2	1	0	9	8	7	6	5	4
4	4	3	2	1	0	9	8	7	6	5
5	5	4	3	2	1	0	9	8	7	6
6	6	5	4	3	2	1	0	9	8	7
7	7	6	5	4	3	2	1	0	9	8
8	8	7	6	5	4	3	2	1	0	9
9	9	8	7	6	5	4	3	2	1	0

UNIDADES

P \ Q	0	1	2	3	4	5	6	7	8	9
0	0	1	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1	1
2	0	0	0	1	1	1	1	1	1	1
3	0	0	0	0	1	1	1	1	1	1
4	0	0	0	0	0	1	1	1	1	1
5	0	0	0	0	0	0	1	1	1	1
6	0	0	0	0	0	0	0	1	1	1
7	0	0	0	0	0	0	0	0	1	1
8	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	0	0	0

DECENAS

P*Q

UNIDADES

P \ Q	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	1	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	1	8	5	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	1

DECENAS

P \ Q	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	1	1	1	1
3	0	0	0	0	1	1	1	2	2	2
4	0	0	0	1	1	2	2	2	3	3
5	0	0	1	1	2	2	3	3	4	4
6	0	0	1	1	2	3	3	4	4	5
7	0	0	1	2	2	3	4	4	5	6
8	0	0	1	2	3	4	4	5	6	7
9	0	0	1	2	3	4	5	6	7	8

30. Aritmética de alta precisión: suma

FN L(AB) averigua cuál de los dos números, A o B, es el mayor, y los dos bucles del 0 al 9 extraen el contenido de la tabla de consulta.

Después de la entrada (INPUT), el programa encuentra la posición del punto decimal e invierte el número (véase 'Resta').

D$[1] se rellena con ceros usando la función STRING$, ya que ésta define la longitud mínima de la cadena necesaria para albergar al número (es igual a la parte entera mayor más la parte decimal mayor más 1).

En las cuatro líneas siguientes, MID$ se usa con su significado de sentencia para reemplazar algunos ceros de D$ por X$ e Y$. La suma se efectúa en SUB 1200 y 1240, empleando el valor (VAL) de P$ y Q$ para encontrar las respuestas correctas. Observe que P$ y Q$ se definen utilizando MID$ como una función para devolver una parte específica de D$.

C es el arrastre, y, en la impresión, (S[RL−L+1]=0) se usa para suprimir los ceros no significativos, si los hubiese.

```
10 REM Aritmetica de alta precision: suma (solo numeros positivos)
20 DEFINT A,B,C,L,P,Q,R,X,Y,Z
30 DEF FN L(A,B)=A-(B-A)*(B>A)
40 DEF FN A$(A)=MID$(STR$(A),2)
50 DIM A$[9,9],B$[9,9]
60 FOR P=0 TO 9: FOR Q=0 TO 9
70 A$[P,Q]=MID$(STR$(100+P+Q),4,1)
80 B$[P,Q]=MID$(STR$(100+P+Q),3,1)
90 NEXT Q,P
100 LINE INPUT "X=";X$: LINE INPUT "Y=";Y$: IF X$="" OR Y$="" OR
LEFT$(X$,1)="-" OR LEFT$(Y$,1)="-" THEN 100
```

```
110 GOSUB 1000: GOSUB 2000
120 ERASE D$,S: END
1000 LX=LEN(X$): LY=LEN(Y$)
1010 XL=INSTR(X$,".")-1: YL=INSTR(Y$,".")-1
1020 IF XL=-1 THEN XL=LX
1030 IF YL=-1 THEN YL=LY
1040 XR=LX-XL+(LX<>XL): YR=LY-YL+(LY<>YL)
1050 L=FN L(XL,YL): R=FN L(XR,YR): RL=R+L
1060 DIM D$[1]: FOR P=0 TO 1
1070 D$[P]=STRING$(RL,48): NEXT
1080 IF XR<>0 THEN MID$(D$[0],1)=RIGHT$(X$,XR)
1090 IF YR<>0 THEN MID$(D$[1],1)=RIGHT$(Y$,YR)
1100 IF XL<>0 THEN MID$(D$[0],RL-XL+1)=LEFT$(X$,XL)
1110 IF YL<>0 THEN MID$(D$[1],RL-YL+1)=LEFT$(Y$,YL)
1120 DIM S[RL+1]: C=0
1130 IF R<>0 THEN FOR P=R TO 1 STEP -1: P$=MID$(D$[0],P,1):
Q$=MID$(D$[1],P,1): GOSUB 1200: NEXT
1150 IF L<>0 THEN FOR P=RL TO RL-L+1 STEP -1: P$=MID$(D$[0],P,1):
Q$=MID$(D$[1],P,1): GOSUB 1240: NEXT: S[RL-L+1]=C
1170 RETURN
1200 S[P]=VAL(A$[VAL(P$),C]): C=VAL(B$[VAL(P$),C]): Z=S[P]
1210 S[P]=VAL(A$[Z,VAL(Q$)]): C=C+VAL(B$[Z,VAL(Q$)])
1220 RETURN
1240 S[P+1]=VAL(A$[VAL(P$),C]): C=VAL(B$[VAL(P$),C]): Z=S[P+1]
1250 S[P+1]=VAL(A$[Z,VAL(Q$)]): C=C+VAL(B$[Z,VAL(Q$)])
1260 RETURN
2000 IF L<>0 THEN FOR P=RL-L+1-(S[RL-L+1]=0) TO RL+1: PRINT FN
A$(S[P]);: NEXT
2010 IF L=0 THEN PRINT FN A$(C);
2020 IF R<>0 THEN PRINT ".";: FOR P=1 TO RL-L: PRINT FN A$(S[P]);: NEXT
2030 RETURN
```

Ejemplo

```
run
X=123456789.987654321
Y=12121212112.1212121
12244558902.108866421
```

31. Aritmética de alta precisión: resta

El primer grupo de líneas rellena la tabla de consulta de la resta. El −20∗(P<Q) transforma el 9 de la resta en un 1 para un arrastre, como en la rutina de la suma (véanse las tablas).

FN L(A,B) define qué número es mayor, A o B, y FN A$(A) se usa para que la impresión de la cadena se lleve a cabo en formato empaquetado.

Después de la entrada (INPUT), con SIGN$ cuidando del signo, el programa encuentra la posición del punto decimal con INSTR(X$,".") y, por resultar más conveniente, se invierten los números (líneas 1080-1110).

Por ejemplo,

```
123.45678
-45.678
```

se convierte en

```
45678123
-67800045
```

después de llenar primero D$ con ceros.

La resta se efectúa en dos bucles separados, usando las rutinas SUB 1160 y PRINT para reconstruir la respuesta con su punto decimal y su signo correctos. Observe el uso de Z en la subrutina para preservar el valor correcto de D[P] en la última sentencia, y no el valor de D[P] procedente de la sentencia 'siguiente-al-último'.

```
10 REM Aritmetica de alta precision: resta (solo numeros positivos)
20 DEFINT A,B,C,L,P,Q,R,X,Y,Z
30 DEF FN L(A,B)=A-(B-A)*(B>A)
40 DEF FN A$(A)=MID$(STR$(A),2)
50 DIM S$[9,9],T$[9,9]
60 FOR P=0 TO 9: FOR Q=0 TO 9
70 S$[P,Q]=MID$(STR$(100+P-Q-20*(P<Q)),4,1)
80 T$[P,Q]=MID$(STR$(100+P-Q-20*(P<Q)),3,1)
90 NEXT Q,P
100 SING$="+": LINE INPUT "X=";X$: LINE INPUT "Y=";Y$: IF X$="" OR
Y$="" OR LEFT$(X$,1)="-" OR LEFT$(Y$,1)="-" THEN 100
110 IF VAL(X$)<VAL(Y$) THEN SW$=X$: X$=Y$: Y$=SW$: SIGN$="-"
120 GOSUB 1000: GOSUB 2000
130 ERASE D$,D: END
1000 LX=LEN(X$): LY=LEN(Y$)
1010 XL=INSTR(X$,".")-1: YL=INSTR(Y$,".")-1
1020 IF XL=-1 THEN XL=LX
1030 IF YL=-1 THEN YL=LY
1040 XR=LX-XL+(LX<>XL): YR=LY-YL+(LY<>YL)
1050 L=FN L(XL,YL): R=FN L(XR,YR): RL=R+L
1060 DIM D$[1]: FOR P=0 TO 1
1070 D$[P]=STRING$(RL,48): NEXT
1080 IF XR<>0 THEN MID$(D$[0],1)=RIGHT$(X$,XR)
1090 IF YR<>0 THEN MID$(D$[1],1)=RIGHT$(Y$,YR)
1100 IF XL<>0 THEN MID$(D$[0],RL-XL+1)=LEFT$(X$,XL)
1110 IF YL<>0 THEN MID$(D$[1],RL-YL+1)=LEFT$(Y$,YL)
1120 DIM D[RL]: C=0
1130 IF R<>0 THEN FOR P=R TO 1 STEP -1: GOSUB 1160: NEXT
```

```
1140 IF L<>0 THEN FOR P=RL TO RL-L+1 STEP -1: GOSUB 1160: NEXT:
1150 RETURN
1160 P$=MID$(D$[0],P,1): Q$=MID$(D$[1],P,1)
1170 D[P]=VAL(S$[VAL(P$),C]): C=VAL(T$[VAL(P$),C]): Z=D[P]
1180 D[P]=VAL(S$[Z,VAL(Q$)]): C=C+VAL(T$[Z,VAL(Q$)])
1190 RETURN
2000 PRINT SIGN$;:IF L<>0 THEN FOR P=1+R TO RL: PRINT FN A$(D[P]);:
NEXT
2010 IF R<>0 THEN PRINT ".";: FOR P=1 TO R: PRINT FN A$(D[P]);: NEXT
2020 RETURN
```

Ejemplo

```
X = 1357913579135779
Y = 24682468.2468246

+1357913333231110.7531754
```

32. Aritmética de alta precisión: multiplicación

La primera parte sigue muy de cerca las rutinas de la suma y de la resta, pero el punto decimal, si es que aparece, se elimina, y no se invierte el número.

Después de la sentencia DIM, la multiplicación se realiza con las tablas de consulta, pero las unidades y las decenas se mantienen separadas y la suma de cada columna se almacena en M[L,1] y M[L,2]. Se debe evitar los espacios en blanco que contiene D$ como resultado de la operación DIM antes de que puedan sumarse las entradas apropiadas en P[L] usando la función VAL.

El 0.00001 se suma para evitar errores de redondeo.

```
10 REM Aritmetica de alta precision: multiplicacion
20 DEFINT A,B,C,L,M,P,Q,X,Y
30 DEF FN A$(A)=MID$(STR$(A),2)
40 DIM M$[9,9],N$[9,9]
50 FOR P=0 TO 9: FOR Q=0 TO 9
60 M$[P,Q]=MID$(STR$(100+P*Q),4,1)
70 N$[P,Q]=MID$(STR$(100+P*Q),3,1)
80 NEXT Q,P
90 LINE INPUT "X=";X$: LINE INPUT "Y=";Y$:
100 IF X$="" OR Y$="" OR LEFT$(X$,1)="-" OR LEFT$(Y$,1)="-" THEN 90
110 GOSUB 1000: GOSUB 2000: END
1000 XD=INSTR(X$,".")-1: YD=INSTR(Y$,".")-1
1010 IF XD=-1 THEN XD=LEN(X$): GOTO 1030
1020 X$=LEFT$(X$,XD)+MID$(X$,XD+2)
```

```
1030 IF YD=-1 THEN YD=LEN(Y$): GOTO 1050
1040 Y$=LEFT$(Y$,YD)+MID$(Y$,YD+2)
1050 LX=LEN(X$): LY=LEN(Y$): L=LX+LY: LD=L-XD-YD
1060 DIM D$[L,LY,1],M[L,2],P[L]
1070 FOR P=1 TO LX: FOR Q=1 TO LY
1080 A=VAL(MID$(X$,P,1)): B=VAL(MID$(Y$,Q,1))
1090 D$[P+Q,Q,0]=M$[A,B]: D$[P+Q-1,Q,1]=N$[A,B]
1100 NEXT Q,P
1110 FOR P=L TO 1 STEP -1: FOR Q=1 TO LY
1120 IF D$[P,Q,0]<>" " THEN M[P,1]=M[P,1]+VAL(D$[P,Q,0])
1130 IF D$[P,Q,1]<>" " THEN M[P,2]=M[P,2]+VAL(D$[P,Q,1])
1140 NEXT Q,P
1150 C=0: FOR P=L TO 1 STEP -1
1160 M=C+M[P,1]+M[P,2]: C=0
1170 IF M>9 THEN C=INT(M/10+0.00001): M=INT(10*(M/10-INT(M/10))+0.0
0001)
1180 P[P]=M: NEXT
1190 RETURN
2000 IF L<>LD THEN FOR P=1-(P[1]=0) TO L-LD: PRINT FN A$(P[P]);: NEXT
2010 IF LD<>0 THEN PRINT ".";: FOR P=L-LD+1 TO L: PRINT FN A$(P[P]);:
NEXT
2020 ERASE D$,M,P: RETURN
```

Ejemplos

```
run
X=1234.56789
Y=98765.4321
121932631.112635269

run
X=999999999999
Y=999999999999
999999999998000000000001
```

33. Aritmética de alta precisión: números recíprocos

Este programa es el más interesante del grupo. Se basa en el hecho de que si A es el recíproco de P, siendo P un entero, obviamente $A=1/P$.

Si A_i es la i-ésima aproximación a $1/P$, el valor exacto de A viene dado por

$$A=A_i+(1/P)*(1-A_iP)$$

Si sustituimos A_i por $1/P$, entonces

$$A_i+1=A_i+A_i(1-A_iP)$$
$$=2A_i-A_i2P$$

Como estamos trabajando con números enteros, tenemos que suprimir el punto decimal, si es que apareciese, multiplicando por una potencia de 10 apropiada. A continuación, el programa calcula el resultado de $200\times A$ (200 corresponde a $2A$ con dos dígitos adicionales) y resta A^2P, lo que da la siguiente aproximación mejorada en dos dígitos.

El primer bloque forma las seis tablas de consulta requeridas.

Una vez que se ha introducido (INPUT) el número como I$, se comprueba su signo y se busca la posición decimal en 200 o 220, dependiendo de si el número es mayor o menor que 1. E$ es la primera aproximación a la respuesta, que es un entero que representa los primeros 8 dígitos distintos de cero del recíproco calculado por el ordenador.

E$ se envia a SUB Por 200 y a continuación a Multiplicación para obtener el cuadrado. Esto se envía más tarde a Multiplicación por C$ para producir A^2P, y luego se efectúa la resta en SUB Resta para hallar el siguiente valor de E$.

Se puede dar más velocidad a la rutina con sólo tener en cuenta lo que ocurre más allá de las posiciones de la cadena donde los dígitos cambian cada vez. Si usted quiere ser un experto en la manipulación de cadenas, averigüe cuáles son las modificaciones necesarias.

Resulta interesante intentar grabar este programa. No ocurre nada durante 40 segundos, tiempo durante el cual el ordenador reorganiza su memoria y se dehace de la información no válida. Esto también ocurre cuando usted escribe FRE(""), lo que da la cantidad de espacio libre para cadenas.

Ejemplo

```
run
X=17
Precision requerida? 20
5.88235294117647058823 6E-2
Ready
```

La introducción de PRINT E$ dentro del bucle I muestra cómo se construye el número.

```
run
X=17
Precision requerida
58823529
5882352942
588235294118
58823529411765
5882352941176471
588235294117647059
58823529411764705883
5.882352941176470588236E-2
```

Observe que el último dígito siempre es alto. También, como ocurre con algunos números primos, pero no con todos, da un decimal periódico que se repite tras un número de posiciones decimales igual al número primo mismo, en este caso 17.

```
10 REM Aritmetica de alta precision: reciprocos
20 DEFINT C,I,K,L,N,P,Q,T,V,Z
30 DEF FN A$(A)=MID$(STR$(A),2)
40 DIM A$[9,9],B$[9,9],M$[9,9],N$[9,9],S$[9,9],T$[9,9]
50 FOR P=0 TO 9: FOR Q=0 TO 9
60 A$[P,Q]=MID$(STR$(100+P+Q),4,1)
70 B$[P,Q]=MID$(STR$(100+P+Q),3,1)
80 M$[P,Q]=MID$(STR$(100+P*Q),4,1)
90 N$[P,Q]=MID$(STR$(100+P*Q),3,1)
100 S$[P,Q]=MID$(STR$(100+P-Q-20*(P<Q)),4,1)
110 T$[P,Q]=MID$(STR$(100+P-Q-20*(P<Q)),3,1)
120 NEXT Q,P
130 T=1: LINE INPUT "X=";I$: IF I$="" THEN 130
140 IF VAL(I$)=0 THEN 130
150 IF LEFT$(I$,1)="+" OR LEFT$(I$,1)="-" THEN T=2
160 ID=INSTR(I$,".")-1: IF ID=-1 THEN ID=LEN(I$): N=0: GOTO 180
170 N=LEN(I$)-1-ID
180 X=ABS(VAL(I$)): IF X=1 THEN GOSUB 2020: END
190 M=X: K=0: IF X<1 THEN K=1: GOTO 220
200 WHILE M>1: M=M/10: K=K+1: WEND
210 GOTO 230
220 WHILE M<1: M=M*10: K=K-1: WEND
230 E$=MID$(STR$(INT(10↑(K+7)/X)),2)
240 C$=MID$(I$,T,ID)+MID$(I$,ID+2)
250 INPUT "Precision requerida"; C: C=ABS(INT(C)): IF C<9 THEN 250
260 GOSUB 1000: GOSUB 2000: END
```

```
1000 FOR I=1 TO C-7 STEP 2
1010 X$=E$: GOSUB 1500: REM Por 200
1020 Y$=X$: GOSUB 1600: REM Multiplicar
1030 X$=Z$: Y$=C$: GOSUB 1600: REM Multiplicar
1040 X$=V$: Y$=LEFT$(Z$,LEN(Z$)-4-K-N-I): GOSUB 1800: REM Restar
1050 NEXT
1060 RETURN
1499 REM Por 200
1500 LX=LEN(X$): DIM S[LX+2]: C=0
1510 FOR P=LX TO 1 STEP -1: VX=VAL(MID$(X$,P,1))
1520 S[P-1]=VAL(A$[VX,C]): C=VAL(B$[VX,C]): Z=S[P-1]
1530 S[P]=VAL(A$[VX,Z]): C=C+VAL(B$[VX,Z])
1540 NEXT: S[0]=C: V$=""
1550 FOR P=0 TO LX+2: V$=V$+FN A$(S[P]): NEXT
1560 ERASE S: RETURN
1599 REM Multiplicar
1600 LX=LEN(X$): LY=LEN(Y$): L=LX+LY
1610 DIM D$[L,LY,1],M[L,2],P[L]
1620 FOR P=1 TO LX: FOR Q=1 TO LY
1630 VX=VAL(MID$(X$,P,1)): VY=VAL(MID$(Y$,Q,1))
1640 D$[P+Q,Q,0]=M$[VX,VY]: D$[P+Q-1,Q,1]=N$[VX,VY]
1650 NEXT Q,P
1660 C=0: FOR P=L TO 1 STEP -1: FOR Q=1 TO LY
1670 IF D$[P,Q,0]<>" " THEN M[P,1]=M[P,1]+VAL(D$[P,Q,0])
1680 IF D$[P,Q,1]<>" " THEN M[P,2]=M[P,2]+VAL(D$[P,Q,1])
1690 NEXT Q,P
1700 C=0: FOR P=L TO 1 STEP -1
1710 M=C+M[P,1]+M[P,2]: C=0
1720 IF M>9 THEN C=INT(M/10+0.00001): M=INT(10*(M/10-C)+0.001)
1730 P[P]=M: NEXT: Z$=""
1740 FOR P=1-(P[1]=0) TO L: Z$=Z$+FN A$(P[P]): NEXT
1750 ERASE D$,M,P: RETURN
1799 REM Restar
1800 LX=LEN(X$): LY=LEN(Y$)
1810 DIM D$[1]: FOR P=0 TO 1
1820 D$[P]=STRING$(LX,48): NEXT
1830 MID$(D$[0],1)=X$: MID$(D$[1],LX-LY+1)=Y$
1840 DIM D[LX]: C=0
1850 FOR P=LX TO 1 STEP -1
1860 VX=VAL(MID$(D$[0],P,1)): VY=VAL(MID$(D$[1],P,1))
1870 D[P]=VAL(S$[VX,C]): C=VAL(T$[VX,C])
1875 Z=D[P]: D[P]=VAL(S$[Z,VY])
1880 C=C+VAL(T$[Z,VY]): NEXT: E$=""
1890 FOR P=1-(D[1]=0) TO LX: E$=E$+FN A$(D[P]): NEXT
1900 ERASE D$,D: RETURN
2000 IF T=2 THEN PRINT LEFT$(I$,1);
```

```
2010 PRINT LEFT$(E$,1);".";MID$(E$,2);"E";-K: RETURN
2020 PRINT I$
2030 RETURN
```

34. Secuencia ordenada (INORDER)

Esta subrutina etiqueta los vértices de un árbol binario en una secuencia ordenada. A cada vértice situado a la izquierda de cualquier otro que esté más cerca de la raíz se le da un valor más bajo, y al que esté a la derecha un valor superior. La ilustración muestra un árbol binario etiquetado en una secuencia ordenada con los nombres de lenguajes de ordenador. (Véase 'Árbol de búsqueda binario' para una rutina relacionada.)

```
10 REM Secuencia ordenada para palabras
20 INPUT "Altura del arbol";K: K=2↑(K+2)-1: L=(K+1)/2-1
30 DIM A$[K],A[K],S[K]
40 FOR P=1 TO L: INPUT A$[P]: NEXT
50 GOSUB 1000: GOSUB 2000: END
1000 C=1: V=1: SP=K
1010 WHILE A$[2*V]<>"": GOSUB 1060: V=V+V: WEND
1020 A[V]=C: C=C+1
1030 IF A$[2*V+1]<>"" THEN V=2*V+1: GOTO 1010
1040 IF SP<>K THEN GOSUB 1070: GOTO 1020
1050 RETURN
1060 S[SP]=V: SP=SP-1: RETURN
1070 V=S[SP+1]: SP=SP+1: RETURN
2000 FOR P=1 TO L: PRINT A$[P];TAB(10);A[P]: NEXT
2010 RETURN
```

```
10 REM Secuencia ordenada para numeros
20 INPUT "Altura del arbol";K: K=2↑(K+2)-1: L=(K+1)/2-1
30 DIM A[K,1],S[K]
40 FOR P=1 TO L: INPUT A[P,0]: NEXT
50 GOSUB 1000: GOSUB 2000: END
1000 C=1: V=1: SP=K
1010 WHILE A[2*V,0]<>0: GOSUB 1060: V=V+V: WEND
1020 A[V,1]=C: C=C+1
1030 IF A[2*V+1,0]<>0 THEN V=2*V+1: GOTO 1010
1040 IF SP<>K THEN GOSUB 1070: GOTO 1020
1050 RETURN
1060 S[SP]=V: SP=SP-1: RETURN
1070 V=S[SP+1]: SP=SP+1: RETURN
2000 FOR P=1 TO L: PRINT A[P,0];TAB(10);A[P,1]: NEXT
2010 RETURN
```

Ejemplo

```
run                      resultado
Altura del arbol? 4      MALLARD     11
? MALLARD                BCPL        4
? BCPL                   QLSUPER     14
? QLSUPER                BASIC09     2
? BASIC09                FORTH       7
? FORTH                  PASCAL      12
? PASCAL                 SPECTRUM    15
? SPECTRUM               ALGOL       1
? ALGOL                  BBC         3
? BBC                    COBOL       6
? COBOL                  LISP        9
? LISP                               0
?                        PIP         13
? PIP                                0
?                                    0
?                                    0
?                                    0
?                                    0
?                                    0
? C                      C           5
?                                    0
? FORTRAN                FORTRAN     8
? LOGO                   LOGO        10
?                                    0
?                                    0
?                                    0
?                                    0
?                                    0
?                                    0
?                                    0
?                                    0
                         Ready
```

Observe que las palabras están numeradas en orden alfabético.

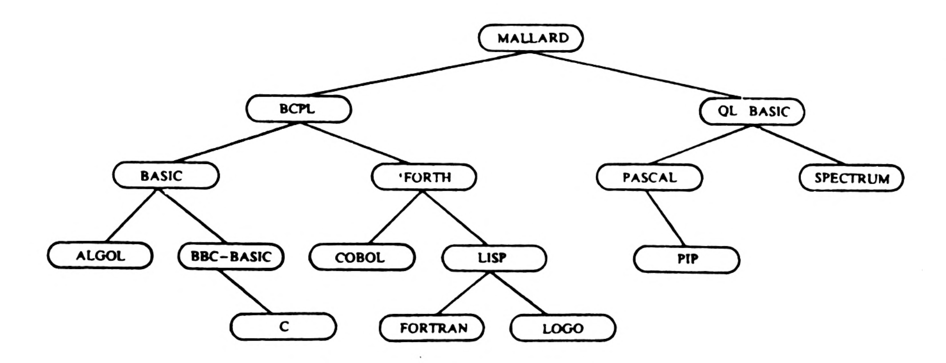


Fig. 34.1. Árbol binario de lenguajes de ordenador.

35. Interpolación

La idea que subyace al término interpolación es ajustar algún tipo de curva a un conjunto de puntos, de forma que usted pueda calcular el valor de una posición intermedia en la que no haya punto. La mejor forma de hacer esto depende mucho de la calidad de los datos que se manejen, y no hay reglas de dificultad o rapidez para obtener la mejor respuesta.

Obviamente, siempre se puede dibujar una línea que atraviese dos puntos, una curva de segundo orden (circunferencia, parábola, elipse o hipérbola) que pase por tres puntos, una curva cúbica a través de cuatro puntos, etc. Sin embargo, no hay garantías para el trazado de una curva de *n*-ésimo orden a través de un conjunto de puntos si la dispersión a partir de una línea recta se debe a errores de medida. Entonces es mejor usar una aproximación de 'La línea mejor ajustada' y emplear su ecuación para predecir el valor y los posibles errores asociados.

De forma similar, si un conjunto de puntos está situado en torno a una curva, es más fácil que ésta puede volver a ser trazada en una escala log/lineal o log/log para producir una banda de dispersión recta que se puede usar después para obtener la aproximación de 'La línea mejor ajustada'.

Si los datos son muy precisos, la forma de encontrar el polinomio que represente la curva es seleccionar un número de puntos espaciados a lo largo de la curva, incluidos el primero y el último del margen de interés, y trazar una curva cuya potencia sea una unidad menor que el número de puntos, es decir, ajustar un polinomio de potencia cuatro si usted necesita cinco puntos, etc.

Esto se demuestra mejor en el ejemplo. La tabla muestra los valores t de Student para $N=10$ a 60. Para obtener una ecuación de tercer grado, necesitamos que

$$aN^3+bN^2+cN+d=t$$

Eligiendo los puntos $N=10$, 15, 25 y 60, tenemos

$$a*1000 \quad +b*100 \quad +c*10+d=2.228$$
$$a*3375 \quad +b*225 \quad +c*15+d=2.131$$
$$a*15625 \quad +b*625 \quad +c*25+d=2.060$$
$$a*216000+b*3600+c*60+d=2.000$$

y resolviéndolas con la subrutina del apartado 'Sistemas de ecuaciones' se obtienen los valores

$a=-0.000014$
$b=0.001520$
$c=-0.050755$
$d=2.597524$

Los valores intermedios que aparecen en la tabla se han calculado a partir de estos valores; la columna *Diferencia* muestra cómo la curva aproximada ondula en las inmediaciones de la línea verdadera. Entre 10 y 27 el error es menos del 0.4%, lo que resulta bastante bueno para nuestro objetivo. Para obtener un ajuste mejor es necesario trazar una curva de orden superior.

Tabla *t* de Student

N	*Real de las tablas*	*Calculado a partir de la ecuación*	*Diferencia (Calculado – Real)*
10	2.228*	(2.228)	–
11	2.201	2.205	+0.004
12	2.179	2.183	+0.004
13	2.160	2.164	+0.004
14	2.145	2.147	+0.002
15	2.131*	(2.131)	–
16	2.120	2.117	–0.003
17	2.110	2.105	–0.005
18	2.101	2.095	–0.006
19	2.093	2.086	–0.007
20	2.086	2.078	–0.008
21	2.080	2.072	–0.008
22	2.074	2.068	–0.006
23	2.069	2.064	–0.005
24	2.064	2.061	–0.003
25	2.060*	(2.060)	–
26	2.056	2.059	+0.003

N	*Real de las tablas*	*Calculado a partir de la ecuación*	*Diferencia (Calculado – Real)*
27	2.052	2.060	+0.008
28	2.048	2.061	+0.013
29	2.045	2.063	+0.018
30	2.042	2.065	+0.026
40	2.021	2.103	+0.082
60	2.000*	(2.000)	–

* Usado para establecer los coeficientes.

Para terminar, una palabra de advertencia. No añada más exactitud a los valores calculados de la que es inherente a los datos originales, y **nunca** trate de calcular valores fuera del intervalo que utilizó en principio.

36. Etiqueta

Esta rutina está relacionada con el almacenamiento de información. Ésta tiene dos partes: un identificador que es único, por ejemplo un nombre o un número, y la información restante relacionada con el identificador, por ejemplo una dirección, un número de teléfono, detalles de los componentes, etc. El identificador se almacena en una matriz (I) y la información restante en otra (J). A cada identificador se asocia una etiqueta que apunta a la segunda matriz, de forma que, por ejemplo, si la información está ordenada, sólo hay que ordenar la matriz del identificador.

```
10 REM Etiqueta
20 PRINT "Cuantos elementos hay que almacenar";: INPUT No.
30 DIM I$[No.],J$[No.]
40 PRINT "Introduzca un identificador unico, por ejemplo un nombre o numero de
parte"
50 FOR P=1 TO No.
60 INPUT "Identificador";I$[P]
70 I$[P]=I$[P]+"*"+STR$(P)
80 INPUT "Restantes detalles";J$[P]
90 NEXT
1000 REM Una busqueda producira el valor V, que contiene la informacion del
identificador"
1010 PRINT I$[V],J$[VAL(MID$(I$[V],INSTR(I$[V],"*")+1))]
```

37-41. Bucles

Los bucles son un medio muy potente de generar y manipular datos. Estos ejemplos ilustran operaciones más complicadas que los bucles sencillos o anidados.

En el 'bucle partido', cuando A es mayor que 4 el punto de retorno del bucle cambia a la segunda sentencia FOR.

Dentro del 'bucle mixto' funcionan dos operaciones de incremento, es decir, FOR A=1 TO 7 y A=A+1, de forma que en vez de 7×4 pares de valores obtenemos 8. El término +(A=4) sirve para ajustar el punto de cambio.

En el 'bucle aleatorio' el valor B salta al superar las diversas condiciones «mayor que».

En el 'bucle circular' la función MOD se usa para cambiar la última entrada por la primera, de forma que el bucle puede realizar operaciones cíclicas. Un ejemplo de esto sería el desplazamiento de todas las entradas de una matriz a través de un número dado de posiciones, usando la posición cero como almacén temporal de los puntos de trazado para obtener una figura cerrada.

Hay ocasiones en que se necesita un conjunto de bucles anidados, dependiendo el número de éstos de alguna variable del programa. Para resolver esto, debería usarse en el bucle una variable con subíndices, por ejemplo FOR A[P]=1 TO B[P], pero la sintaxis de Amstrad no lo permite. Sin embargo, se puede subsanar este problema simulando las operaciones efectuadas a lo largo del bucle, pero sin usar una sentencia FOR...NEXT principal. Se puede encontrar ejemplos de esta construcción en los apartados 'Permutación', 'Anagrama' y 'Árboles 2-3'.

En el ejemplo dado, que es la preparación de una tabla de consulta para la multiplicación de P*Q (P y Q varían de 1 a 10), se compara una estructura de bucle de tamaño variable con un bucle doble. Éste se puede incrementar fácilmente a un sistema P*Q*R, pero no es posible conseguir un bucle anidado sencillo sin escribir un bucle adicional en el programa en la variable R.

37. Bucle partido

```
10 REM Bucle partido
20 FOR A=1 TO 5:IF A>4 THEN FOR A=13 TO 16:PRINT A:NEXT:GOTO 40
30 PRINT A
40 NEXT
```

Este programa selecciona valores de A iguales a 1, 2, 3, 4, 13, 14, 15, 16.

38. Bucle mixto

```
10 REM Bucle mixto
20 FOR A=1 TO 7: FOR B=5 TO 8
30 PRINT TAB(18);A;CHR$(8);",";B
40 A=A+1+(A=4): NEXT B,A
```

Este programa selecciona pares de valores de la forma siguiente:

1,5;2,6;3,7;4,8;5,5;6,6;7,7;8,8

39. Bucle aleatorio

```
10 REM Bucle aleatorio
20 FOR a=1 TO 4: FOR b=1 TO 2
30 IF b>1 THEN FOR b=7 TO 9: IF b>8 THEN FOR b=11 TO 15 STEP 2: PRINT b;:
NEXT: GOTO 40 ELSE PRINT b;: GOTO 40 ELSE PRINT b;: GOTO 50
40 NEXT
50 NEXT: PRINT
60 NEXTs
```

Este programa imprime los números 1, 7, 8, 11, 13 y 15 cuatro veces.

40. Bucle circular

Este bucle selecciona puntos adyacentes en una matriz, de forma que, por ejemplo, cuando sean imprimidos produzcan una figura cerrada. (Véase el apartado 'Tipos de ángulos'.)

```
10 REM Bucle circular
20 INPUT N
30 DIM A[N],B[N]
40 FOR P=1 TO N: A[P]=P: B[P]=P: NEXT
50 GOSUB 1000: END
1000 FOR P=0 TO N
1010 A[P]=A[(1+P)MOD(N+1)]
1020 NEXT
1030 FOR P=1 TO N: PRINT "Unir";B[P];"a";A[P]: NEXT
1040 RETURN
```

41. Bucle anidado de tamaño variable

```
10 REM Bucle anidado de tamaño variable
20 INPUT N:REM Numero de bucles
```

```
30 DIM A[N]:REM Variables del bucle
40 DIM B[N]:REM Valores finales del bucle
50 FOR P=1 TO N
60 A[P]=1: REM Inicializar variables
70 B[P]=10: REM Inicializar valores finales del bucle
80 NEXT P
90 FOR P=1 TO N
100 IF A[P]>B[P] THEN A[P]=1: REM Reinicializar variables del bucle interno si
es necesario
110 NEXT P: Z=1
120 FOR P=1 TO N
130 Z=Z*A[P]: NEXT P: REM Calcular producto
140 PRINT Z;
150 FOR X=N TO 1 STEP -1: REM Operacion de anidamiento de control
160 A[X]=A[X]+1: REM Incrementar variable
170 IF A[X]>B[X] THEN PRINT: NEXT X: PRINT: END: REM Cambiar al bucle
siguiente?
180 GOTO 90: REM Siguiente ciclo de este bucle
```

Sin la sentencia REM la rutina queda así:

```
10 REM Bucle anidado de tamaño variable
20 INPUT N: DIM A[N],B[N]
30 FOR P=1 TO N: A[P]=1: B[P]=10: NEXT
40 FOR P=1 TO N: IF A[P]>B[P] THEN A[P]=1
50 NEXT: Z=1
60 FOR P=1 TO N: Z=Z*A[P]: NEXT: PRINT Z;
70 FOR X=N TO 1 STEP -1: A[X]=A[X]+1
80 IF A[X]>B[X] THEN PRINT: NEXT ELSE 40: END
```

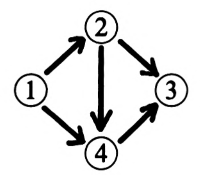
Esta subrutina es de gran valor para los problemas que implican permutaciones de las variables.



42-48. Matrices

Las matrices se pueden usar para almacenar información sobre trayectorias entre lugares de mapas o gráficos y entre las esquinas de un polígono o un poliedro sólido. Por ejemplo, la matriz usada en la 'rotación universal' emplea "1" para indicar qué esquinas de una pirámide doble están unidas y "0" como indicador de que no se da esta unión.

Los puntos 1 a 4 se pueden unir por varias trayectorias, como muestra el dibujo.



La matriz de trayectorias sería entonces

$$P = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

P^2 tiene la propiedad de representar trayectorias formadas por dos partes separadas.

Por tanto

$$P^2 = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

El 2 representa 1→3 como 1→2→3 y 1→4→3
Los 1 representan 1→4 como 1→2→4 y 2→3 como 2→4→3

$$P^3 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

es decir, 1→2→4→3, la única trayectoria triple.

P^4 es una matriz nula, ya que no hay trayectorias de cuatro partes.

Consulte las secciones 'Sistemas de ecuaciones', 'Determinantes' y 'Regresión' para encontrar otras aplicaciones de las matrices.

La multiplicación de matrices se efectúa multiplicando las filas por las columnas. Por ejemplo

$$\begin{bmatrix} A & B \\ E & F \end{bmatrix} * \begin{bmatrix} C & D \\ G & H \end{bmatrix} = \begin{bmatrix} A*C+B*G & A*D+B*H \\ E*C+F*G & E*D+F*H \end{bmatrix}$$

Por tanto, una matriz M×N sólo se puede multiplicar por una matriz N×R para dar un resultado M×R, o bien se puede multiplicar una matriz A×B×C por una C×D para obtener una respuesta A×B×D. En todo caso, dos de las dimensiones deben ser iguales y desaparecer del producto resultante. El orden de multiplicación es, por tanto, importante, y en general A*B no es lo mismo que B*A.

El siguiente ejemplo ilustra la multiplicación de matrices usando costes de salarios, pero también podrían ser niveles de existencias y valores de situaciones diferentes, o resultados de exámenes de distintos sujetos en años diferentes, etc.

El número de empleados de cada nivel en seis establecimientos diferentes es el siguiente:

ESTABLECIMIENTO	NIVEL				
	1	2	3	4	5
A	200	50	10	1	0
B	250	60	6	2	0
C	500	101	12	3	0
D	35	5	1	0	1
E	1010	190	27	2	1
F	1250	200	45	3	1

y las escalas de salarios son:

NIVEL	*Libras/año*
1	5500
2	7500
3	9000
4	11000
5	15000

Entonces

$$\begin{bmatrix} 200 & 50 & 10 & 1 & 0 \\ 250 & 60 & 6 & 2 & 0 \\ 500 & 101 & 12 & 3 & 0 \\ 35 & 5 & 1 & 0 & 1 \\ 1010 & 190 & 27 & 2 & 1 \\ 1250 & 200 & 45 & 3 & 1 \end{bmatrix} * \begin{bmatrix} 55000 \\ 7500 \\ 9000 \\ 11000 \\ 15000 \end{bmatrix} = \begin{bmatrix} 1576000 \\ 1901000 \\ 3648500 \\ 254000 \\ 7260000 \\ 8828000 \end{bmatrix}$$

representado simbólicamente por

A[M,N]*B[N,R]=C[M,R]

da la cuenta total de los salarios de los seis establecimentos.

Una matriz unitaria es el equivalente de un número 1 ordinario y consiste en una línea diagonal de unos, siendo el resto de las entradas ceros. El producto de una matriz por su inversa es una matriz unitaria. El apartado 'Rotación universal' contiene un ejemplo de una matriz unitaria de 3×3, establecida inicialmente para constituir la matriz de transformación final.

Las matrices complejas son en realidad pares de matrices que contienen las partes real e imaginaria de la matriz de valores. Tienen usos muy variados; un ejemplo sencillo es el de la rotación alrededor del origen. La expresión

$$(x+iy)\exp(i\theta)$$

produce la rotación del punto (x,y) en un ángulo θ alrededor del origen.

Como $\exp(i\theta)=\cos\theta+i\operatorname{sen}\theta$, tenemos, con múltiples puntos y ángulos, el producto de dos matrices, es decir:

$$\begin{bmatrix} X1 + iY1 \\ X2 + iY2 \\ X3 + iY3 \\ X4 + iY4 \end{bmatrix} * \quad \cos\theta + i \operatorname{sen}\theta \ \cos\theta + i \operatorname{sen}\theta \ \cos\psi + i \operatorname{sen}\psi$$

Necesitará la inversa de una matriz en la solución de un sistema de ecuaciones y para comprobar si un determinante es cero o no. La inversa análoga de un par de matrices complejas también viene dada junto con el conjugado complejo que se necesita ocasionalmente.

42. Multiplicación de matrices

```
10 REM Multiplicacion de matrices
20 DEFINT J,K,L,M,N,R
```

```
30 DEF FN N$(N)=MID$(STR$(N),2)
40 INPUT "Numero de filas de la matriz A";M
50 INPUT "Numero de columnas de la matriz A";N
60 INPUT "Numero de columnas de la matriz B";R
70 DIM A[M,N],B[N,R],C[M,R]
80 FOR J=1 TO M: FOR K=1 TO N
90 PRINT "A(";FN N$(J);",";FN N$(K);")=";: INPUT A[J,K]: NEXT K,J
100 FOR J=1 TO N: FOR K=1 TO R
110 PRINT "B(";FN N$(J);",";FN N$(K);")=";: INPUT B[J,K]: NEXT K,J
120 REM O use la rutina de introduccion de datos (matrices) dos veces para
introducir A y B)
130 GOSUB 1000: CLS: GOSUB 2000: END
1000 FOR J=1 TO M: FOR L=1 TO R
1010 FOR K=1 TO N: C[J,L]=C[J,L]+A[J,K]*B[K,L]
1020 NEXT K,L,J
1030 RETURN
2000 FOR J=1 TO M: FOR K=1 TO R
2010 LOCATE 4*K-3,2*J+3: PRINT C[J,K]
2020 NEXT K,J
2030 RETURN
2040 REM O use la rutina de impresion de matrices
```

Ejemplo

```
run
Numero de filas en la matriz A? 3
Numero de columnas en la matriz A? 4
Numero de columnas en la matriz B? 2
A(1,1)=? 1
A(1,2)=? 2
A(1,3)=? 3
A(1,4)=? 4
A(2,1)=? 5
A(2,2)=? 6
A(2,3)=? 7
A(2,4)=? 8
A(3,1)=? 9
A(3,2)=? 10
A(3,3)=? 11
A(3,4)=? 12
B(1,1)=? 2
B(1,2)=? 4
B(2,1)=? 6
B(2,2)=? 8
B(3,1)=? 10
B(3,2)=? 12
B(4,1)=? 14

100 120
228 280
356 440
Ready
```

43. Multiplicación de matrices complejas

```
10 REM Multiplicacion de matrices complejas
20 DEFINT J,K,L,M,N,P,R
30 DEF FN N$(N)=MID$(STR$(N),2)
40 INPUT "Numero de filas de la matriz A";M
50 INPUT "Numero de columnas de la matriz A";N
60 INPUT "Numero de columnas de la matriz B";R
70 CLS: P=1
80 DIM A[M,N],B[N,R],C[M,R],X[M,N],Y[N,R],Z[M,R]
90 FOR J=1 TO M: FOR K=1 TO N
100 P=P+1: LOCATE 1,P: PRINT "A(";FN N$(J);",";FN N$(K);")=";: INPUT
A[J,K]
110 LOCATE 20,P: PRINT "+i*X(";FN N$(J);",";FN N$(K);")=";: INPUT X[J,K]:
NEXT K,J
120 FOR J=1 TO N: FOR K=1 TO R
130 P=P+1: LOCATE 1,P: PRINT "B(";FN N$(J);",";FN N$(K);")=";: INPUT
B[J,K]
140 LOCATE 16,P: PRINT "+i*Y(";FN N$(J);",";FN N$(K);")=";: INPUT Y[J,K]
150 NEXT K: NEXT J
160 REM O use la rutina de introduccion de datos (matrices) dos veces para
introducir A y B)
180 GOSUB 1000: CLS: GOSUB 2000: END
1000 FOR J=1 TO M: FOR L=1 TO R: FOR K=1 TO N
1010 C[J,L]=C[J,L]+A[J,K]*B[K,L]-X[J,K]*Y[K,L]
1020 Z[J,L]=Z[J,L]+A[J,K]*Y[K,L]-X[J,K]*B[K,L]
1030 NEXT K,L,J
1040 RETURN
2000 FOR J=1 TO M: FOR K=1 TO N
2010 LOCATE 8*K-3,2*J+3: PRINT C[J,K];"+i*";Z[J,K]
2020 NEXT K,J
2030 RETURN
2040 REM O use la rutina de impresion de matrices
```

44. Inversión de matrices

La división de dos matrices no es posible, pero se puede derivar A^{-1} y multiplicar usando las reglas de multiplicación de matrices, dado que A es cuadrada y su determinante no es cero.

Esta derivación de A^{-1} se basa en el método de sustitución progresiva de Gauss-Jordan, en el que se establecen dos bucles y se trabaja sobre las entradas individuales por orden, siendo reemplazados inmediatamente los valores de la matriz por valores nuevos. Por tanto, en el caso de una matriz de 3×3 hay 27 operaciones, de forma que la matriz se convierte gradualmente en su inversa en 27 pasos.

Relacionando esto con las ecuaciones, si, por ejemplo,

$$Y_1 = a_{11}X_1 + a_{12}X_2 + a_{13}X_3$$
$$Y_2 = a_{21}X_1 + a_{22}X_2 + a_{23}X_3$$
$$Y_3 = a_{31}X_1 + A_{32}X_2 + A_{33}X_3$$

resolviendo la primera ecuación para X_1 y sustituyendo se obtiene (si $a_{11}<>0$)

$$X_1 = A_{11}Y_1 + A_{12}X_2 + A_{13}X_3$$
$$Y_2 = A_{21}Y_1 + A_{22}X_2 + A_{23}X_3$$
$$Y_3 = A_{31}Y_1 + A_{32}X_2 + A_{33}X_3$$

donde

$$A_{11} = 1/a_{11} \qquad A_{12} = -a_{12}/a_{11} \qquad A_{13} = -a_{13}/a_{11}$$
$$A_{21} = a_{21}/a_{11} \qquad A_{22} = a_{22} - a_{21}a_{12}/a_{11} \qquad A_{23} = a_{23} - a_{21}a_{13}/a_{11}$$
$$A_{31} = a_{31}/a_{11} \qquad A_{32} = a_{32} - a_{31}a_{12}/a_{11} \qquad A_{33} = a_{33} - a_{31}a_{13}/a_{11}$$

La repetición de este proceso a través de los tres ciclos tiene como resultado la matriz inversa.

SUB 1120 busca ceros en la diagonal y, si es posible, intercambia dos columnas para evitar el problema de la división por cero al principio del bucle I. La matriz M recuerda qué columnas han sido intercambiadas, y en SUB 1160 cambia las mismas filas a la inversa para compensar la primera modificación.

Si dos filas, o dos columnas, son idénticas, o una es múltiplo de la otra, o si aparece una línea de ceros, la matriz no puede ser invertida, ya que su determinante es cero. De establecer esto se encarga la sentencia IF del comienzo de la rutina.

La sentencia IF J<>N+(I=N) del bucle K se encarga de la necesidad de un segundo conjunto de bucles que aparece en algunas versiones de esta rutina.

```
10 REM Inversion de matrices
20 DEFINT I,J,K,N,T: DEF FN N$(N)=MID$(STR$(N),2): N=0
30 WHILE N<=0: INPUT "N=";N: WEND
40 DIM A[N,N]: FOR I=1 TO N: FOR J=1 TO N
50 PRINT "A(";FN N$(I);",";FN N$(J);")=";: INPUT A[I,J]: NEXT J,I
60 GOSUB 1000: GOSUB 2000: END
1000 DIM M[N,N]: T=0: GOSUB 1120
1010 FOR I=1 TO N: IF ABS(A[I,I])<0.0000001 THEN T=1: RETURN
1020 A[I,I]=1/A[I,I]
1030 FOR J=1 TO N: IF J=I THEN 1100
1040 A[J,I]=A[J,I]*A[I,I]
1050 FOR K=1 TO N: IF K=I THEN 1090
1060 A[J,K]=A[J,K]-A[J,I]*A[I,K]
1070 IF J<>N+(I=N) THEN 1090
1080 A[I,K]=-A[I,I]*A[I,K]
```

```
1090 NEXT K
1100 NEXT J,I: IF T THEN GOSUB 1160
1110 RETURN
1120 FOR I=1 TO N: IF A[I,I]=0 THEN T=-1: GOSUB 1140
1130 NEXT I: RETURN
1140 FOR J=1 TO N: IF A[I,J]<>0 THEN FOR K=1 TO N: SW=A[K,I]:
A[K,I]=A[K,J]: A[K,J]=SW: NEXT K: M[I,J]=1: RETURN
1150 NEXT J: RETURN
1160 FOR I=1 TO N: FOR J=1 TO N
1170 IF M[I,J]=1 THEN FOR K=1 TO N: SW=A[I,K]: A[I,KI]=A[J,K]:
A[J,K]=SW: NEXT K
1180 NEXT J,I: RETURN
2000 ERASE M: IF T<1 THEN FOR I=1 TO N: FOR J=1 TO N: PRINT "A(";FN
N$(I);",";FN N$(J);")=";A[I,J]: NEXT J,I: RETURN
2010 PRINT "Matriz singular; no tiene inversa":RETURN
```

Ejemplo

$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ 1 & 0 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} -1.51.125 & & .25 \\ 0 & .25 & -.5 \\ & .5-.375 & .25 \end{bmatrix}$$

45. Inversión de matrices complejas

```
10 REM Inversion de matrices complejas
20 DEFINT A,B,N,J,K,T
30 DEF FN M(A,B)=R[A,B]*R[A,B]+I[A,B]
40 DEF FN Z$(A$,A,BOOLE)=MID$(A$,1+A,-LEN(A$)*BOOLE)
50 DEF FN N$(N)=MID$(STR$(N),2)
60 INPUT "Introduzca el rango de la matriz";N$
70 IF N$="" THEN 60
80 T=1: IF LEFT$(N$,1)="+" THEN T=2
90 FOR P=T TO LEN(N$): IF MID$(N$,P)<"0" OR MID$(N$,P)>"9" THEN 60
100 NEXT: N=VAL(N$):  IF N<=0 THEN 60
110 DIM R[N,N],I[N,N]
120 PRINT "   Ahora introduzca los terminos de la matriz, que pueden ser reales,
imaginarios o complejos. Normalmente tendran la forma a+ib, donde i=SQR(-1).
Escriba a y b. Pulse RETURN para continuar.": INPUT K$
130 CLS: T=1
140 FOR I=1 TO N: FOR J=1 TO N: T=T+1
150 LOCATE 1,T: PRINT "a";FN N$(I);FN N$(J);" es ";: INPUT R(I,J)
160 LOCATE 20,T: PRINT "+i*";;: INPUT I(I,J)
170 NEXT J,I
180 GOSUB 1000: GOSUB 2000: END
```

```
1000 DIM M[N,N]: T=0: GOSUB 1120
1010 FOR I=1 TO N: Z=FN M(I,I): IF ABS(Z)<0.00000001 THEN T=1: GOTO 2000
1020 R[I,I]=R[I,I]/Z: I[I,I]=-I[I,I]/Z
1030 FOR J=1 TO N: IF J=1 THEN 1100
1040 Z=R[J,I]: R[J,I]=Z*R[I,I]-I[J,I]*I[I,I]: I[J,I]=I[J,I]*R[I,I]+Z*I[I,I]
1050 FOR K=1 TO N: IF K=I THEN 1090
1060 R[J,K]=R[J,K]-R[J,I]*R[I,K]+I[J,I]*I[I,K]: I[J,K]=I[J,K]-I[J,I]*
R[I,K]-R[J,I]*I[I,K]
1070 IF J<>N+(I=N) THEN 1090
1080 Z=R[I,K]: R[I,K]=-R[I,K]*Z+I[I,I]*I[I,K]: I[I,K]=-I[I,I]*Z-R[I,I]*I[I,K]
1090 NEXT
1100 NEXT J,I: IF T THEN GOSUB 1160
1110 RETURN
1120 FOR I=1 TO N: IF FN M(I,I)=0 THEN T=-1: GOSUB 1140
1130 NEXT: RETURN
1140 FOR J=1 TO N: IF FN M(I,J)<>0 THEN FOR K=1 TO N: SW=R[K,I]:
R[K,I]=R[K,J]: R[K,J]=SW: SW=I[K,I]: I[K,I]=I[K,J]: I[K,J]=SW: M[I,J]=1:
NEXT: RETURN
1150 NEXT: RETURN
1160 FOR I=1 TO N: FOR J=1 TO N:
1170 IF M[I,J]=1 THEN FOR K=1 TO N: SW=R[I,K]: R[I,K]=R[J,K]:
R[J,K]=SW: SW=I[I,K]: I[I,K]=I[J,K]: I[J,K]=SW: NEXT
1180 NEXT J,I: RETURN
1190 RETURN
2000 ERASE M: IF T>0 THEN PRINT "Matriz singular; no tiene inversa.": RETURN
2005 PRINT "La matriz inversa es":PRINT
2010 FOR I=1 TO N: FOR J=1 TO N: GOSUB 2050
2020 PRINT "b(";FN N$(I);",";FN N$(J);")=";Z$
2030 NEXT J,I
2040 RETURN
2050 X$=FN Z$("-",0,R[I,J]<0)+FN Z$(STR$(R[I,J]),1,R[I,J]<>0)+FN
Z$("+",0,ABS(R[I,J])>0 AND I[I,J]>0)+FN Z$("-",0,I[I,J]<0)
2060 Y$=FN Z$(STR$(I[I,J]),1,I[I,J]<>0 AND ABS(I[I,J])<>1)+FN Z$
("i",0,I[I,J]<>0)+FN Z$("0",0,ABS(R[I,J])<0.000001 AND ABS(I[I,J])<0.000001)
2070 Z$=X$+Y$
2080 RETURN
```

Ejemplo

Introduzca el rango de la matriz ? 3.
Ahora introduzca los terminos individuales de la
matriz, que pueden ser reales, imaginarios
o complejos. Habitualmente tendran la
forma a+bi, donde i=SQR(-1).
Escriba a y b. Pulse RETURN para continuar

```
a11 es ?        1 +i*? 0
a12 es ?        0 +i*? 0
a13 es ?        0 +i*? 5
a21 es ?        0 +i*? -2
a22 es ?        2 +i*? 0
a23 es ?        0 +i*? 0
a31 es ?        1 +i*? 0
a32 es ?        1 +i*? 1
a33 es ?        0 +i*? 0

B11 = 0
B12 = -.5+.5i
B13 = -i
B21 = 0
B22 = -.5i
B23 = 1
B31 = -.2i
B32 = -.1-.1i
B33 = .2
```

46. Matriz unitaria

DIM se usa para crear los ceros, y el bucle se encarga de la diagonal de unos.

```
10 REM Matriz unitaria
20 DEFINT I,J,N
30 INPUT "Rango de la matriz";N: IF N<1 THEN 30
40 GOSUB 1000: GOSUB 2000: END
1000 DIM A[N,N]
1010 FOR I=1 TO N: A[I,I]=1: NEXT
1020 RETURN
2000 FOR I=1 TO N: FOR J=1 TO N
2010 PRINT A[I,J];: NEXT: PRINT: NEXT
2020 RETURN
```

47. Transposición

Al llevar a cabo la transposición de una matriz cuadrada, las filas pasan a ser columnas y viceversa.

```
10 REM Trasposicion de una matriz
20 DEFINT J,K,N
30 INPUT "Rango de la matriz";N: IF N<1 THEN 30
40 DIM A[N,N]
```

```
50 FOR J=1 TO N: FOR K=1 TO N: INPUT A[J,K]: NEXT K,J
60 GOSUB 1000: GOSUB 2000: END
1000 FOR J=1 TO N: FOR K=J TO N
1010 IF J<>K THEN SW=A[J,K]: A[J,K]=A[K,J]: A[K,J]=SW
1020 NEXT: NEXT
1030 RETURN
2000 FOR J=1 TO N: FOR K=1 TO N
2010 PRINT A[J,K];: NEXT K: PRINT: NEXT J
2020 RETURN
```

48. Conjugada compleja

La conjugada compleja tiene el signo contrario en las partes imaginarias.

```
10 REM Conjugada compleja de un par de matrices complejas
20 DEFINT J,K,N
30 INPUT "Rango de la matriz";N: IF N<1 THEN 30
40 DIM REAL [N,N],IMAG [N,N]
50 FOR J=1 TO N: FOR K=1 TO N: INPUT REAL [J,K],IMAG [J,K]: NEXT K,J:
REM Introduzca REAL [J,K] e IMAG [J,K] como dos numeros reales separados por
una coma
60 GOSUB 1000: GOSUB 2000: END
1000 FOR J=1 TO N: FOR K=1 TO N
1010 IMAG [J,K]=-IMAG [J,K]
1020 NEXT: NEXT
1030 RETURN
2000 FOR J=1 TO N: FOR K=1 TO N
2010 PRINT REAL [J,K];IMAG [J,K];"i": NEXT K,J
2020 RETURN
```

49. Menú

El objetivo de un menú en un programa es ayudar al usuario a interactuar con el ordenador y a poder elegir una parte específica del programa sin tener que ejecutarlo entero. Los programas complicados tienen a menudo menús de ayuda a los que usted puede volver si llega a un punto en el que no sabe qué hacer.

El siguiente programa ilustra la formación de un menú usando ON...GOSUB como elemento activo. Si detiene el programa en cualquier punto y escribe GOTO 120, el menú aparecerá en la pantalla. Normalmente, las subrutinas volverán a la línea 120 una vez finalizadas; la última de ellas le permitirá terminar el programa. Otras subrutinas se encargan de la grabación o de la carga de los datos.

El punto interesante del programa es la instrucción POKE de la segunda línea, que altera CHR$(208) para conseguir un subrayado mejor. M$[10] contiene los títulos de los capítulos, y éstos son impresos por el bucle P.

La elección se hace como M$[0], cuyas características numéricas son comprobadas en SUB 300; de lo contrario FAIL se convierte en verdadero y se vuelve a solicitar la elección. Parte de la dificultad de este área está en impedir que el menú sea desplazado de la pantalla por los diversos mensajes, de forma que, en lo posible, la pantalla esté reservada para entradas activas. Sin embargo, si usted introduce una coma accidentalmente, no puede impedir la aparición del mensaje 'Redo from start'.

```
10 REM Menu
20 SYMBOL AFTER 208: H=HIMEM+1: POKE H+1,0
30 WINDOW #1,1,40,22,25: DIM M$[10]
40 NL$=CHR$(10)+CHR$(13)
50 M$[1]="  1.  Cargar datos"
60 M$[2]="  2.  Grabar datos"
70 M$[3]="  3.  Tercera opcion"
80 M$[4]="  4.  Cuarta opcion"
90 REM etc.
100 M$[9]="  9.  Novena eleccion"
110 M$[10]="  10.  Fin"
120 CLS: LOCATE 18,2: PRINT "MENU"
130 LOCATE 18,3: FOR P=1 TO 4: PRINT CHR$(208);: NEXT: PRINT
140 FOR P=1 TO 10: LOCATE 5,4+P
150 PRINT M$[P]: NEXT: T$[0]=""
160 PRINT #1,"Escriba su eleccion 1-10"
170 T$=INKEY$: IF T$="" THEN 170
180 PRINT #1,T$;: IF T$<>CHR$(13) THEN T$[0]=T$[0]+T$: GOTO 170
190 GOSUB 300: IF NOT FAIL THEN M%=VAL(T$[0]) ELSE 170
200 IF M%>10 OR M%<1 THEN 170
210 ON M% GOSUB 400,500,600,700,800,900,1000,1100,1200,1310: REM O lo que
sea apropiado
220 GOTO 120
240 END
300 FAIL=0
310 FOR P=1 TO LEN(M$[0]): Z$=MID$(M$[0],P,1)
320 IF Z$<"0" OR Z$>"9" THEN FAIL=-1
330 NEXT: RETURN
400 REM Cargar los datos del magnetofono
410 CLS: PRINT "  Esta preparado el magnetofono para introducir los datos
almacenados (s/n)?";NL$
420 IF INKEY$<>"S" AND INKEY$<>"s" THEN 420
430 OPENIN D$
440 INPUT #9,A,B,C: REM Aqui A,B,C corresponden a las variables A,B,C, cuyos
valores ha almacenado usted previamente
450 CLOSEIN
460 RETURN
500 REM Grabar datos
510 CLS: PRINT "Esta preparado el magnetofono para grabar los datos.
```

```
(s/n)";NL$;" Compruebe la posicion de la cinta."
520 IF INKEY$<>"s" AND INKEY$<>"S" THEN 520
530 PRINT "Escriba el nombre del fichero";: INPUT D$
540 OPENOUT D$
550 PRINT #9,A,B,C: REM Aqui A,B,C corresponden a los nombres de las variables
cuyos valores quiere usted grabar. Vease 'Arbol 2-3'
560 CLOSEOUT
570 RETURN
1300 REM Fin
1310 CLS: PRINT "De verdad desea acabar?";NL$
1320 PRINT "Para acabar pulse ";CHR$(34);"S";CHR$(34);" y despues
<INTRO>";: INPUT K$
1330 K$=UPPER$(K$): IF K$="S" THEN NEW ELSE RETURN
```

50. Mezcla

Esta subrutina mezcla dos listas ya ordenadas en una sola, ordenada también. Si usted necesita añadir valores adicionales a una lista ordenada, resulta mucho más eficaz ordenarlos primero y después mezclar las dos listas. Lleva mucho más tiempo introducirlos individualmente en los lugares correctos o añadirlos todos al final de la lista para volver a ordenarlos después.

```
10 REM Mezcla
20 PRINT "Longitud de las listas.";" Introduzca los valores como dos numeros sepa-
rados por una coma"
25 DEF FN A$(A)=MID$(STR$(A),2)
30 INPUT M,N: IF M=0 OR N=0 THEN 30
40 M=M+1: N=N+1: TITCH=0
50 DIM A[M],B[N],C[M+N]
60 FOR P=2 TO M: PRINT "A(";FN A$(P-1);")=";: INPUT A[P]
70 IF A[P]<TITCH THEN TITCH=A[P]
80 NEXT
90 FOR P=2 TO N: PRINT "B(";FN A$(P-1);")=";: INPUT B[P]
100 IF B[P]<TITCH THEN TITCH=B[P]
110 NEXT
120 GOSUB 1000: GOSUB 2000: END
1000 A[1]=TITCH-1: B[1]=A[1]
1010 Y=M: Z=N
1020 WHILE A[Y]>B[Z]: C[Y+Z]=A[Y]: IF Y>1 THEN Y=Y-1: WEND
1030 C[Y+Z]=B[Z]: IF Z>1 THEN Z=Z-1: GOTO 1020
1040 RETURN
2000 FOR P=3 TO M+N: PRINT C[P];: NEXT
2010 ERASE A,B,C: RETURN
```

51. MIN/MAX

Esta subrutina produce el mismo resultado que MIN y MAX, pero es mucho más flexible. La subrutina encuentra los valores máximo y mínimo de una lista. Funciona ordenando los valores introducidos contiguamente y seleccionando el máximo y el mínimo de cada par. Después se comparan estos valores en grupos de 4, 8, 16, etc., seleccionando el máximo y el mínimo en cada caso.

```
10 REM Min/Max
20 DEFINT A,B,N,P,S: N=0
30 DEF FN A$(A)=MID$(STR$(A),2)
40 WHILE N<=0: INPUT "Numero de valores";N: WEND
50 GOSUB 1000: GOSUB 2000: END
1000 M=N: P=0
1010 WHILE M>1: M=M/2: P=P+1: WEND
1020 S=2↑P: DIM A[S]
1030 FOR A=1 TO N: PRINT "A(";FN A$(A);")=";: INPUT A[A]
1040 A=A+1: IF A>N THEN 1080
1050 PRINT "A(";FN A$(A);")=";: INPUT A[A]
1060 IF A[A]>A[A-1] THEN SW=A[A]: A[A]=A[A-1]: A[A-1]=SW
1070 NEXT
1080 IF M<>1 THEN FOR A=N+1 TO S: A[A]=A[N]: NEXT
1090 M=2
1100 FOR B=0 TO S/2/M-1: T=2*B*M
1110 IF A[1+T]<A[1+T+M] THEN A[1+T]=A[1+T+M]
1120 IF A[T+M]<A[T+M+M] THEN A[T+M+M]=A[T+M]
1130 NEXT: M=M+M: IF M<S THEN 1100
1140 RETURN
2000 PRINT : PRINT "Min=";;A[S];SPACE$(5);"Max=";A[1]
```

52. Mínimo, máximo, media, mediana y moda

Es más fácil encontrar el valor mínimo y el máximo colocando los números en orden ascendente. La media se obtiene a partir de la suma de los valores durante la fase de entrada. La mediana se sitúa justo en la mitad de la lista, y será uno de los valores de ésta si hay un número impar de valores, o bien será la media de dos valores adyacentes si el número total de valores es par.

La moda es más difícil de obtener, ya que tiene usted que contar el número de entradas de cada valor. T[0,Z] se encarga de esto, y T[1,Z] almacena los valores actuales diferentes. En E, se ordena T[0,Z] de mayor a menor por el procedimiento de clasificación por burbujas; la moda corresponde al primer valor de T[1,Z]. En la clasificación se intercambian T[0,Z] y T[1,Z].

Esta subrutina imprime los valores mencionados de una lista de números positivos.

```
10 REM Minimo, maximo, media, mediana y moda
20 DEF FN A$(A)=MID$(STR$(A),2)
30 INPUT "Numero de valores";N: IF N<=0 THEN 30
40 DIM V[N]: S=0
50 FOR P=1 TO N: PRINT "V(";FN A$(P);")=";: INPUT V[P]: S=S+V[P]: NEXT
60 GOSUB 1000: GOSUB 2000: END
1000 Q=0: FOR P=1 TO N-1
1010 IF V[P+1]<V[P] THEN SW=V[P]: V[P]=V[P+1]: V[P+1]=SW: Q=Q+1
1020 NEXT: IF Q<>0 THEN 1000
1030 P=1: Z=1: DIM T[1,N]
1040 IF V[P]=V[P+1] THEN T[0,Z]=T[0,Z]+1: P=P+1: IF P<N THEN 1040
1050 T[1,Z]=V[P]: P=P+1: Z=Z+1+T[0,Z]: IF P<N THEN 1040
1060 Q=0: FOR P=1 TO N-1
1070 X=P+1: IF T[0,X]>T[0,P] THEN SW=T[0,P]: T[0,P]=T[0,X]:T[0,X]=SW:
SW=T[1,P]: T[1,P]=T[1,X]: T[1,X]=SW: Q=Q+1
1080 NEXT: IF Q<>0 THEN 1060
1090 RETURN
2000 PRINT "Minimo =";V[1]: PRINT
2010 PRINT "Maximo =";V[N]: PRINT
2020 PRINT "Media =";S/N: PRINT
2030 IF 2*INT(N/2)=N THEN PRINT "Mediana=";(V[N/2]+V[N/2+1])/2 ELSE
PRINT "Mediana=";V[N/2]
2040 PRINT: PRINT "Moda    =";T[1,1]: PRINT
2050 ERASE T: RETURN
```

53. *Módulo*

A pesar de que raramente se necesita un signo de módulo, esta rutina está incluida como ejemplo del uso de POKE para alterar un carácter ASCII. El apartado 'Fichero de imagen' explica los valores empleados con POKE.

```
10 REM Modulo
20 GOSUB 1000: GOSUB 2000: END
1000 SYMBOL AFTER 209
1010 H=HIMEM+1
1020 FOR P=H TO H+7
1030 POKE P,4: POKE P+16,64
1040 NEXT
1050 RETURN
2000 PRINT CHR$(209);"M";CHR$(211);"es el modulo de M"
2010 RETURN
```

54. Filtro de nombres

Esta subrutina permite el almacenamiento de los nombres. Son caracteres válidos las mayúsculas o las minúsculas, los guiones, los puntos, los espacios y los apóstrofos. Las palabras que contienen otros caracteres son rechazadas. El bucle en P comprueba los caracteres normales encontrados en los nombres, pero, si la prueba falla, Q se reduce en uno para no dejar un blanco en B$[N].

```
10 REM Filtro de nombres
20 INPUT "Numero de nombres";N: IF N<1 THEN 20
30 DIM B$[N]
40 GOSUB 1000: GOSUB 2000: END
1000 FOR Q=1 TO N: PRINT "Nombre";Q;" es ";: INPUT A$
1010 FOR P=1 TO LEN(A$): Z$=MID$(A$,P,1)
1020 IF Z$="-" OR Z$="." OR Z$=" " OR Z$="'" OR Z$>="A" AND Z$<="Z"
OR Z$>="a" AND Z$<="z" THEN NEXT: B$[Q]=A$: GOTO 1040
1030 PRINT A$;" no es un nombre": Q=Q-1
1040 NEXT Q
1050 RETURN
2000 FOR P=1 TO N: PRINT B$[P]: NEXT
2010 RETURN
```

Ejemplo

```
run
Numero de nombres? 13
Nombre  1  es  ? J. Lucas-Tooth
Nombre  2  es  ? T. Harding
Nombre  3  es  ? Sr. 5. Smith
Sr. 5 Smith no es un nombre
Nombre  3  es  ? Sr. R. Smith
Nombre  4  es  ? I. O'Connel
Nombre  5  es  ? Sr. P. Ward-Jones
Nombre  6  es  ? Rachelina
Nombre  7  es  ? Johnson W.
Nombre  8  es  ? Clive Bosworth
Nombre  9  es  ? Srta. J. Robinson
Nombre  10 es  ? T;K. Jones
T;K. Jones no es un nombre
Nombre  10 es  ? T.K. Jones
Nombre  11 es  ? J. Bu'Lock
Nombre  12 es  ? Alison
Nombre  13 es  ? William Peterson
Ready
```

```
(Resultado)
J. Lucas Tooth
T. Harding
Sr. R. Smith
I. O'Connel
Sr.P. Ward-Jones
Rachelina
Johnson W.
Clive Bosworth
Srta. J. Robinson
T.K. Jones
J. Bu'Lock
Alison
William Peterson
```

55. Permutación

Hay n! (n factorial) permutaciones de n elementos; algunos programas necesitan la habilidad para generarlas. La subrutina que aparece a continuación se usa en los apartados 'Anagrama', 'Determinanates por el desarrollo de Laplace' y 'Árboles 2−3', por ejemplo.

Las permutaciones de un conjunto de caracteres se pueden llevar a cabo comenzando por un carácter y añadiendo el siguiente en cada posición posible, como ilustra la figura 1. La subrutina dada funciona de forma ligeramente distinta, rotando progresivamente los caracteres hacia la izquierda (o hacia la derecha, si usted comienza por el extremo opuesto) como se ve en la figura 2. Cuando una segunda rotación lleva a la restauración de una secuencia, el resultado es ignorado.

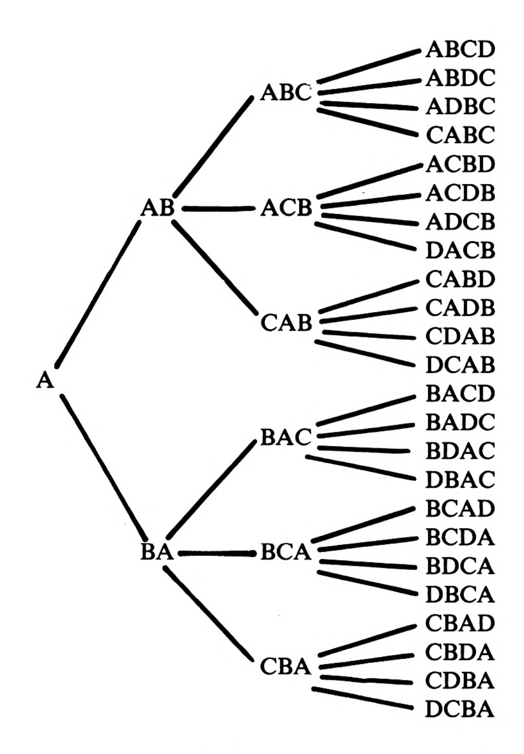


Fig. 55.1. Permutaciones por introducción secuencial del elemento siguiente.

			IMPRESIÓN
Inicio	ABCD	→	ABCD
Rotación desde 2	BACD	→	BACD
2	ABCD		
3	BCAD	→	BCAD
2	CBAD	→	CBAD
2	BCAD		
3	CABD	→	CABD
2	ACBD	→	ACBD
2	CABD		
3	ABCD		
4	BCDA	→	BCDA
2	CBDA	→	CBDA
2	BCDA		
3	CDBA	→	CDBA
2	DCBA	→	DCBA
2	CDBA		
3	DBCA	→	DBCA
2	BDCA	→	BDCA
2	DBCA		
3	BCDA		
4	CDAB	→	CDAB
2	DCAB	→	DCAB
2	CDAB		
3	DACB	→	DACB
2	ADCB	→	ADCB
2	DACB		
3	ACDB	→	ACDB
2	CADB	→	CADB
2	ACDB		
3	CDAB		
4	DABC	→	DABC
2	ADBC	→	ADBC
2	DABC		
3	ABDC	→	ABDC
2	BADC	→	BADC
2	ABDC		
3	BDAC	→	BDAC
2	DBAC	→	DBAC
2	BDAC		
3	DABC		
4	ABCD	→	Fin

Fig. 55.2. Secuencia de permutación de la subrutina.

El siguiente programa (para cuatro caracteres) ejecuta la operación de rotación.

```
10 REM Permutacion simple
20 INPUT A$: REM  LEN(A$)=4
30 FOR R=1 TO 4
40 FOR Q=1 TO 3
50 FOR P=1 TO 2
60 PRINT A$;" ";
70 N=1: GOSUB 110: NEXT P
80 N=2: GOSUB 110: NEXT Q
90 N=3: GOSUB 110: NEXT R
100 END
110 Z$=LEFT$(A$,1)
120 FOR X=1 TO N
130 MID$(A$,X,1)=MID$(A$,X+1,1)
140 NEXT X
160 MID$(A$,X)=Z$
170 RETURN
```

Este programa reclama un bucle externo que le permitiría permutar cualquier número de caracteres, pero esto implicaría el uso de una variable con subíndices, que no es válida como variable de bucle. En lugar de esto, se emplea una estructura de bucle sin FOR. . .NEXT (véase 'Bucles'), cuyo resultado es la siguiente subrutina. Las variables de A[L] se convierten en las variables del bucle y las de B[L] en los valores finales de los bucles (2, 3, 4, etc.). La línea 2 inicializa los valores iniciales y finales de los bucles. La línea 1020 reinicializa A[X] y se la compara con su valor final.

La subrutina imprime todas las permutaciones de los caracteres de una cadena.

```
10 REM Permutacion universal
20 INPUT A$: IF A$="" OR LEN(A$)=1 THEN 20
30 GOSUB 1000: END
1000 L=LEN(A$)-1: DIM A[L],B[L]
1010 FOR P=1 TO L: A[P]=1: B[P]=L-P+2: NEXT
1020 FOR P=1 TO L: IF A[P]>B[P] THEN A[P]=1
1030 NEXT
1040 PRINT A$+SPACE$(7-L);
1050 FOR X=L TO 1 STEP -1
1060 Z$=LEFT$(A$,1): FOR P=1 TO LEN(A$)-X
1070 MID$(A$,P,1)=MID$(A$,P+1,1): NEXT
1080 MID$(A$,P,1)=Z$
1090 A[X]=A[X]+1: IF A[X]>B[X] THEN NEXT: GOTO 1100 ELSE 1020
1100 ERASE A,B: RETURN
```

Ejemplo

```
Primeras 100 permutaciones de 'teaser'
teaser    etaser    eatser    aetser    ateser
taeser    easter    aester    aseter    saeter
```

seater	esater	asteer	sateer	staeer
tsaeer	taseer	atseer	steaer	tseaer
tesaer	etsaer	estaer	setaer	easetr
aesetr	seeatr	eeastr	eeastr	eaestr
aeestr	aeestr	eaestr	aseter	saeter
seater	esater	easter	aester	setaer
estaer	etsaer	tesaer	tseaer	steaer
etaser	teaser	taeser	ateser	aetser
eatser	taseer	atseer	asteer	sateer
staeer	tsaeer	setear	estear	etsear
tesear	tseear	steear	etesar	teesar
teesar	etesar	eetsar	eetsar	tesear
etsear	estear	setear	steear	tseear
esetar	seetar	seetar	esetar	eestar
eestar	eteasr	teeasr	eteasr	eteasr
eetasr	eetasr	teaesr	etaesr	eatesr

56. Permutaciones de tres números

Ésta es una subrutina simplificada de permutación para sólo tres números, desarrollada para efectuar las permutaciones de los índices h, k y l de las caras en el programa del cristal.

Como el primer índice se puede escoger de tres formas, el segundo sólo de dos (porque ya hemos elegido una) y el tercero es fijo, el número total de permutaciones es 3*2*1=6 (n! para n valores).

En el bucle A la elección comienza con

h k l

El procedimiento es intercambiar dos términos dependiendo del valor de T, que es 2 (A par) o 1 (A impar), y así la secuencia de cambios es la siguiente

A	A[0]	A[1]	A[2]	
0	h	k	l	intercambia 0 y 2
1	l	k	h	intercambia 0 y 1
2	k	l	h	intercambia 0 y 2
3	h	l	k	intercambia 0 y 1
4	l	h	k	intercambia 0 y 2
5	k	h	l	intercambia 0 y 1
6	h	k	l	

La subrutina produce las seis permutaciones de tres números cualesquiera h, k y l.

```
10 REM Permutaciones de tres numeros
20 DIM A[2],P[6,2]
```

```
30 INPUT "h=";A[0]
40 INPUT "k=";A[1]
50 INPUT "l=";A[2]
60 GOSUB 1000: GOSUB 2000: END
1000 FOR A=0 TO 6
1010 FOR B=0 TO 2: P[A,B]=A[B]: NEXT
1020 SW=A[0]: A[0]=A[T]: A[T]=SW: NEXT
1030 RETURN
2000 FOR A=0 TO 6: FOR B=0 TO 2
2010 PRINT P[A,B];",";: NEXT: PRINT CHR$(8);CHR$(32);
2020 PRINT: NEXT
2030 RETURN
```

57. La inflación de la postguerra

Los datos para este sencillo cálculo se han tomado del London and Cambridge Economic Bulletin Index y, desde 1974, del Department of Employment Gazette.

La rutina muestra el valor del dinero entre 1945 y 1985 y su repercusión en los precios.

```
10 REM Inflacion de la postguerra
20 DATA 1.00,.945,.9125,.8458,.8209,.8,.7333,.6708,.65,.6375,.6125,.5833,.5625,
.5458,.5417,.5375,.5208,.4958,.4917,.4708,.4417,.4333
30 DATA .425,.4083,.3875,.3583,.33,.31,.28,.2158,.1799,.1458,.1252,.1139,.1041,
.088,.0778,.0695,.0662,.0630,.0559
40 DEFREAL I: DEFINT P: DEFSTR A,V,Y
50 DIM I[41]
60 FOR P=1 TO 41: READ I[P]: NEXT
70 PRINT "Introduzca el valor y el año de la compra (1945-1985)": PRINT: PRINT
"Pta";;: LINE INPUT V: LOCATE POS(#0)+LEN(V)+1,VPOS(#0)-1: PRINT " en
19";: LINE INPUT Y
80 PRINT: PRINT "En que año le gustaria a usted saber su valor? 19";: LINE INPUT
A
90 GOSUB 2000: END
2000 PRINT: PRINT "Costaba #";V;" en 19";Y
2010 V!=VAL(V)
2020 PRINT: PRINT "Su valor en 19";A;" era Pta"; USING "#####.##";V!*I[VAL
(A)-44]/I[VAL(Y)-44]
2030 PRINT: PRINT "Costaria Pta"; USING "#####.##";V!*I[VAL(Y)-44)/I[VAL
(A)-44];: PRINT " comprar en 19";A
2040 PRINT: PRINT "El factor es "; USING "##.####";I[VAL(Y)-44]/I[VAL(A)
-44];: PRINT " a uno"
2050 RETURN
```

58. Números primos

En este apartado encontrará dos subrutinas; una es más rápida que la otra, pero también necesita más memoria.

La primera rutina obtiene los números primos eliminando todos los números que tengan un factor. Tomando como primos el 1, el 2 y el 3, sin cálculos, el programa suma 2 y trata de averiguar si 5 tiene algún factor. Como no es así, se concluye que 5 es un número primo y se pasa al 7, y así sucesivamente. Las divisiones por factores se realizan hasta la raíz cuadrada del número aproximadamente (línea 2060), ya que no puede haber un factor mayor sin que usted haya encontrado un factor menor que la raíz cuadrada.

Una variante de esta rutina almacena los números primos, empleando sin embargo menos memoria que la tercera subrutina, que es más rápida y usa el tamiz de Eratóstenes.

Esta técnica establece una matriz que contiene todos los números impares desde el 3 hasta n. A través del tamiz sólo pasa el número más pequeño, y los múltiplos de éste son eliminados de la lista. Después pasa el menor de los números que quedan y se suprime sus múltiplos, y así sucesivamente.

La tabla da una comparación de las necesidades de tiempo y memoria de las dos subrutinas.

Esta subrutina genera los números primos hasta N.

```
10 REM Numeros primos por division
20 INPUT "N=";N: N=INT(ABS(N))
30 GOSUB 2000: END
2000 A=3
2010 IF N>=1 THEN PRINT 1
2020 IF N>=2 THEN PRINT 2
2030 IF N>=3 THEN PRINT A
2040 B=3: A=A+2
2050 IF A-B*INT(A/B)=0 THEN 2040
2060 IF B>=INT(A/B) THEN 2080
2070 B=B+2: GOTO 2050
2080 IF A<=N THEN PRINT A
2090 IF A>=N THEN RETURN ELSE 2040
```

Esta subrutina genera y almacena los números primos hasta SQR(N) (la raíz cuadrada de N) y también calcula el número de ellos.

```
10 REM Numeros primos hasta la raiz de N, con almacenamiento
20 INPUT "N=";N
30 N=INT(ABS(N)): RT=SQR(N)
40 GOSUB 1000: GOSUB 2000: END
```

```
1000 A=3: Z=4: DIM N[RT/2+2]
1010 IF N>=1 THEN N[1]=1
1020 IF N>=4 THEN N[2]=2
1030 IF N>=9 THEN N[3]=3
1040 B=3: A=A+2
1050 IF A-B*INT(A/B)=0 THEN 1040
1060 IF B>=INT(A/B) THEN 1080
1070 B=B+2: GOTO 1050
1080 IF A<=RT THEN N[Z]=A: Z=Z+1
1090 IF A>=RT THEN RETURN ELSE 1040
2000 FOR P=1 TO RT/2+2: IF N[P]<>0 THEN PRINT N[P]
2010 NEXT
2020 PRINT "El numero de primos hasta SQR";N;"(";RT;") es";Z-1
2030 ERASE N: RETURN
```

```
10 REM Tamiz de Eratostenes
20 DEFINT N,P
30 PRINT "Numero maximo (>3) que hay que considerar": INPUT N
40 IF N<=2 THEN 30
50 GOSUB 1000: GOSUB 2000: END
1000 N=INT((N+1)/2): DIM A[N]: A[1]=3
1010 FOR P=1 TO N-1: A[P+1]=A[P]+2: NEXT
1020 FOR P=1 TO N
1030 NEXT : CLS
1040 RETURN
1050 FOR Q=P TO N-A[P] STEP A[P]: IF Q+A[P]<=N THEN A[Q+A[P]]=0
1060 NEXT : RETURN
2000 PRINT 1: PRINT 2
2010 FOR P=1 TO N-1: IF A[P]<>0 THEN PRINT A[P]
2020 NEXT: RETURN
```

Comparación del tiempo y la memoria usados

Número	*División por factores*		*Tamiz de Eratóstenes*	
	Tiempo (segs.)	*Memoria*	*Tiempo (segs.)*	*Memoria*
10	.27	720	.34	842
20	.55	720	.55	867
50	.96	720	1.05	942
100	2.45	720	1.89	1067
200	5.91	720	3.99	1317
500	17.12	720	9.99	2067
1000	39.81	720	19.55	3317
2000	97.81	720	39.81	5817
5000	325.99	720	99.55	13317

59. Impresión de una matriz o un determinante

En matrices grandes, si los números no son enteros puede que sea necesario redondearlos a cuatro cifras significativas para evitar la sobreimpresión.

La subrutina muestra en pantalla matrices de hasta nueve filas y doce columnas con los corchetes de matriz o de determinante.

```
10 REM Impresion de una matriz o un determinante
15 CLS: INPUT "Modo";S
20 INPUT "M=";M: INPUT "N=";N: IF M<1 OR N<1 THEN 20
30 DIM A[M,N]: FOR J=1 TO M: FOR K=1 TO N
40 INPUT A[J,K]: NEXT K,J
50 GOSUB 2000: END
2000 MODE 2: TAG
2010 MOVE 120,390: PRINT "La matriz es";: REM O "El determinante es"
2020 PLOT 120,374: DRAWR 208/S,0: REM O PLOT 120,374: DRAWR 288/S,0
2030 FOR J=1 TO M: FOR K=1 TO N
2040 MOVE 96*K-40,350-32*J: PRINT A(J,K);: NEXT K,J
2050 PLOT 72,326: DRAWR -20,0: DRAWR 0,4-32*M: DRAWR 20,0
2060 PLOT 96*N+64/S,326: DRAWR 20,0: DRAWR 0,4-32*M: DRAWR -20,0
2070 TAGOFF: RETURN
2080 REM Para los determinantes use PLOT 52,326: DRAWR 0,4-32*M y PLOT
96*N+84/S,326: DRAWR 0,4-32*M en las lineas 2050 y 2060 respectivamente
```

60. Proyección

Se supone que el origen 3-D está cerca del centro de la pantalla del televisor, en el punto (320,196), y la relación de proyección es

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} -cos\alpha & cos\beta & 0 \\ -sin\alpha & -sin\beta & 1 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 320 \\ 196 \\ 0 \end{bmatrix}$$

Fijando la relación de aspecto de la pantalla en 0.92*, las longitudes de los ejes son

Oz=162 pixels
Oy=176.2 pixels
Ox=61.39 pixels

* El valor exacto de la relación de aspecto puede diferir de este valor si las amplificaciones de X e Y en la pantalla del televisor están establecidas de forma diferente. Esto debería ser determinado de forma experimental dibujando un verdadero cuadrado simétricamente en la pantalla.

y los ángulos vienen dados por

$tan\alpha = 1/3$
$tan\beta = 1/27$

Así pues, la relación es

$$X = 176.2y\text{COS(ARCTAN(1/27))} - 61.39x\text{COS(ARCTAN(1/3))} + 320$$
$$Y = 162z - 176.2y\text{SEN(ARCTAN(1/27))} - 61.39x\text{SEN(ARCTAN(1/3))} + 196$$

o

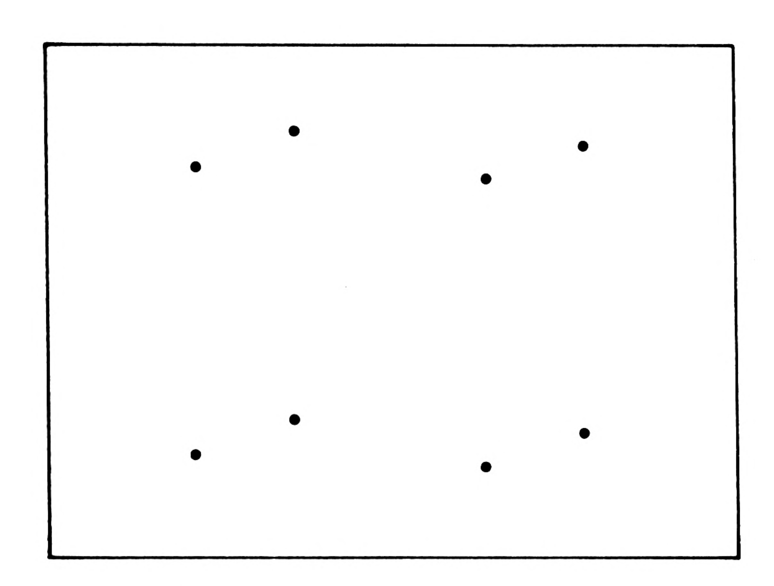
$$X = 320 - 58.2x + 176.1y$$
$$Y = 196.4x - 6.5y + 162z$$

x, y y z se miden a partir de O y X e Y de la parte inferior izquierda de la pantalla. Ox, Oy y Oz se toman como una unidad de longitud.

Ejemplo

El siguiente programa ilustra el uso de estas relaciones, pero puede encontrar otros ejemplos en los apartados 'Cristales cúbicos' y 'Rotación universal'. El programa sencillamente traza los extremos proyectados de un cubo de lado 2.

```
10 REM Proyeccion
20 INPUT "Ampliacion (<1)";M
30 DEF FN X(X,Y,M)=320-(58.24*X-176.08*Y)*M
40 DEF FN Y(X,Y,Z,M)=196-(19.41*X+6.52*Y-162*Z)*M
50 DIM A[8,3]
60 A[1,1]=1: A[1,2]=1: A[1,3]=1
70 A[2,1]=1: A[2,2]=1: A[2,3]=-1
80 A[3,1]=1: A[3,2]=-1: A[3,3]=1
90 A[4,1]=1: A[4,2]=-1: A[4,3]=-1
100 A[5,1]=-1: A[5,2]=-1: A[5,3]=-1
110 A[6,1]=-1: A[6,2]=-1: A[6,3]=1
120 A[7,1]=-1: A[7,2]=1: A[7,3]=-1
130 A[8,1]=-1: A[8,2]=1: A[8,3]=1
140 CLS
150 FOR P=1 TO 8
160 PLOT FN X(A[P,1],A[P,2],M),FN Y(A[P,1],A[P,2],A[P,3],M)
170 NEXT
180 END
```



61. Adentro y afuera

Estas subrutinas relacionadas almacenan valores en una pila usando el principio de 'último en entrar, primero en salir'. El puntero de pila, SP, se usa para indicar qué valor está en la parte superior de la pila, pero los valores reales no son transferidos. La subrutina se emplea en el programa del apartado 'Secuencia ordenada' para almacenar los vértices ocupados a lo largo de la trayectoria hacia un vértice vacío.

```
10 REM Adentro y afuera
20 INPUT K: DIM D[K]: REM V se refiere a algun elemento de la matriz D
30 DIM S[K]: SP=K
40 REM Adentro
50 S[SP]=V: SP=SP-1: RETURN
60 REM Afuera
70 V=S[SP+1]: SP=SP+1: RETURN
```

62. Números enteros pitagóricos

Esta subrutina extrae los pares de números menores que un valor dado, cuyos cuadrados, cuando se los suma, dan un cuadrado perfecto, por ejemplo 3 y 4.

Para evitar elegir un mismo número dos veces, por ejemplo 3 y 4 y 4 y 3, el bucle de B comienza en A. Otro punto interesante es la necesidad de sumar una pequeñísima cantidad para cubrir las inexactitudes de los cálculos. A pesar de que en la pantalla se podría escribir un número como entero, al efectuar una comparación a dicho número hay que restarle uno y sumarle .9999999999 etc. (en binario) e INT redondea hacia abajo, no pudiendo, por tanto, extraer la igualdad en 'IF ABS (T−INT(T))=0'.

```
10 REM Numeros enteros pitagoricos
20 INPUT "Valor maximo que hay que considerar";N
30 FOR A=1 TO N: FOR B=A TO N
40 T=SQR(A*A+B*B)
50 IF ABS (T-INT(T+0.00001))<0.00001 THEN PRINT TAB(2);A;TAB(10);B;
TAB(20);ROUND(T,0)
60 NEXT B: NEXT A
```

63. Solución de una ecuación de segundo grado

Ésta es la solución obvia de una ecuación de segundo grado. Halla las soluciones, tanto reales como imaginarias, de una ecuación de segundo grado del tipo $ax^2+bx+c=0$.

```
10 REM Solucion de la ecuacion de segundo grado
20 NL$=CHR$(10)+CHR$(13)
30 INPUT "a=";A
40 INPUT "b=";B
50 INPUT "c=";C
60 IF A=0 AND B=0 THEN 30
70 GOSUB 1000: GOSUB 2000: END
1000 IF A=0 THEN B$="unica": X=-C/B: RETURN
1010 P=-B/2/A: Q=(B*B-4*A*C)/4/A/A
1020 IF Q>=0 THEN B$="real(es)": X1=P+SQR(Q): X2=P-X1+P: RETURN
1030 B$="imaginaria(s)": R$=STR$(P): I$=STR$(SQR(ABS(Q)))+"*i"
1040 RETURN
2000 CLS: Z$=LEFT$(B$,1): PRINT "La(s) raiz(ces) "; B$;
2010 IF Z$="u" THEN PRINT " es ";X
2020 IF Z$="r" THEN PRINT " son ";NL$;X1;NL$;"y";NL$;X2
2030 IF Z$="i" THEN PRINT " son ";NL$;R$+"+"+I$;NL$;"y";NL$;R$+"-"+I$
2040 RETURN
```

Ejemplos

```
run
a=? 0
b=? 5
c=? 2
La(s) raiz(ces) unica es -0,4
Ready
run
a=? 1
b=? -5
c=? 6
```

```
La(s) raiz(ces) real(es) son
3
2
Ready
run
a=? 1
b=? 8
c=? 20
La(s) raiz(ces) imaginaria(s) son
-4+ 2*i
-4- 2*i
Ready
```

64. Regresión

Un sistema de ecuaciones simultáneas linealmente independientes sólo se puede resolver si el número de ecuaciones es exactamente el mismo que el número de variables desconocidas. Si hay menos ecuaciones que variables, sólo se puede encontrar las relaciones entre las razones de las variables, pero si hay más ecuaciones que variables, es posible obtener una solución única que minimiza el error de las ecuaciones por el método de los 'mínimos cuadrados'.

La subrutina que damos aquí se basa en el método de eliminación de Bauer, que emplea combinaciones ponderadas de las filas. La matriz A es copiada en la matriz U con el fin de protegerla, y la matriz Q, que es de tipo triangular superior derecha, se obtiene por descomposición de U y usando un proceso de ortogonalización con combinaciones de las filas ponderadas. La respuesta, obtenida por sustitución, se almacena en la matriz X.

Muy complicado para explicar en unas cuantas líneas.

Si M y N son iguales, la rutina resuelve el sistema de ecuaciones normalmente, dando una solución exacta.

```
10 REM Solucion de N ecuaciones con M incognitas (M<=N)
20 DEFINT H,I,J,L,M,N
30 DEF FN A$(A)=MID$(STR$(A),2)
40 INPUT "Numero de ecuaciones";N: IF N<=0 THEN 40
50 INPUT "Numero de incognitas";M: IF M>N THEN 40
60 DIM A[N,M],B[N],X[M]
70 FOR I=1 TO N: FOR J=1 TO M
80 PRINT "A(";FN A$(I);",";FN A$(J);")=";: INPUT A[I,J]: NEXT
90 PRINT "B(";FN A$(I);")=";: INPUT B[I]: NEXT
100 GOSUB 1000: GOSUB 2000: END
1000 DIM Q[M*(M+1)/2],U[N,M]
1010 FOR I=1 TO N: FOR J=1 TO M
1020 U[I,J]=A[I,J]: NEXT J,I
```

```
1030 L=0: FOR I=1 TO M
1040 S=0: FOR J=1 TO N: S=S+U[J,I]*U[J,I]: NEXT
1050 L=L+1: Q[L]=S
1060 T=0: FOR J=1 TO N: T=T+U[J,I]*B[J]: NEXT
1070 X[I]=T
1080 FOR H=I+1 TO M: GOSUB 1150: NEXT
1090 NEXT
1100 FOR I=M TO 1 STEP -1
1110 H=L-I: S=X[I]
1120 FOR J=I+1 TO M: S=S-Q[J+H]*X[J]: NEXT
1130 X[I]=S/Q[L]: L=L+I-M-2
1140 NEXT: RETURN
1150 T=0: FOR J=1 TO N: T=T+U[J,I]*U[J,H]: NEXT
1160 L=L+1: Q[L]=T: T=T/S
1170 FOR J=1 TO N: U[J,H]=U[J,H]-U[J,I]*T: NEXT
1180 RETURN
2000 FOR I=1 TO M: PRINT "X(";FN A$(I);")=";X[I]: NEXT
2010 ERASE Q,U: RETURN
```

Ejemplo

La regresión técnica se puede aplicar cuando en un grupo de factores cada uno de ellos tiene un efecto lineal sobre el valor de alguna medida. Por ejemplo, la resistencia del acero a la tracción depende en gran parte del contenido de carbón, pero no por entero, de forma que los demás elementos presentes tienen un efecto menor que se puede expresar por una fórmula de este tipo:

Resistencia a la tracción $= a*$ %-de-carbón $+ b*$ %-de-manganeso $+ c*$ %-de-silicio $+ \ldots$

Para obtener los coeficientes *a*, *b*, *c*, etc. de esta ecuación mediante una regresión técnica, usted necesita un gran número de aceros de composición y resistencia a la tracción conocidas, y luego, empleando la subrutina anterior, puede resolver el sistema de ecuaciones simultáneas para obtener el mejor ajuste entre la resistencia calculada y la medida.

Otro ejemplo ilustrado con detalle es el de la realización de análisis químicos usando los rayos X característicos producidos cuando se irradia una superficie con un rayo de electrones o con rayos X. En una primera aproximación, la señal medida es proporcional a la concentración del elemento, pero los demás elementos interfieren y es posible establecer una ecuación que dé una relación lineal que represente los efectos.

La tabla muestra el análisis de un grupo de aceros inoxidables y las intensidades medidas de los rayos X de cromo. Estos resultados se han tomado de un informe publicado por la British Iron and Steel Research Association (que ahora forma parte de la British Steel Corporation).

Análisis de los aceros inoxidables

(a) Usados para obtener los coeficientes de regresión

Cr %	*Ni %*	*Nb %*	*Ti %*	*Mo %*	*Intensidad de cromo, I*	*Cr %/I−11.61*
4.94	0	0	0	0	0.391	1.024
9.99	0	0	0	0	0.723	2.207
14.82	0	0	0	0	0.999	3.225
19.80	0	0	0	0	1.244	4.306
24.91	0	0	0	0	1.470	5.336
18.22	8.17	0	0	0	1.141	4.358
18.48	6.16	0	0	0	1.161	4.307
20.45	8.19	0	0	0	1.251	4.737
15.29	8.36	0	0	0	0.991	3.819
16.95	10.15	0	0	0	1.075	4.157
18.52	8.03	0.54	0	0	1.139	4.650
18.24	7.97	1.06	0	0	1.109	4.837
18.15	7.88	1.64	0	0	1.096	4.950
18.12	7.81	2.11	0	0	1.080	5.168
18.48	7.98	0	0.43	0	1.140	4.601
18.42	8.00	0	0.95	0	1.117	4.881
18.28	7.92	0	1.39	0	1.095	5.084
18.33	7.83	0	1.97	0	1.084	5.300
18.25	7.93	0	0	0.99	1.122	4.656
18.15	7.85	0	0	2.13	1.088	5.072
17.86	7.77	0	0	3.28	1.045	5.481
17.76	7.63	0	0	4.31	1.019	5.819
17.49	7.54	0	0	5.54	0.83	6.182

(b) Usados como prueba independiente

Cr %	*Ni %*	*Nb %*	*Ti %*	*Mo %*	*Intensidad de cromo, I*	*Porcentaje calculado de Cr*
15.2	6.26	0	0	0	0.992	15.09
12.8	12.45	0	0	0	0.833	12.50
18.7	9.49	0	0	0	1.154	18.61

Cr %	*Ni %*	*Nb %*	*Ti %*	*Mo %*	*Intensidad de cromo, I*	*Porcentaje calculado de Cr*
18.5	9.33	0	0	0	1.143	18.38
18.41	8.97	0	0	0	1.1555	18.53
19.93	7.46	0	0	0	1.228	20.01
16.70	10.15	0	0	0	1.055	16.60
16.18	9.24	0	0	0	1.024	15.95
18.24	8.00	0	0	0	1.134	18.09
18.56	10.18	0	0	0	1.158	18.68
18.10	7.86	0	0	0	1.126	17.92
18.04	8.28	0	0	0	1.131	18.01
19.74	9.16	0	0	0	1.205	19.68
17.80	9.52	1.02	0	0	1.083	17.71
18.45	9.47	0	0.46	0	1.124	18.33
17.60	9.48	0	0	2.43	1.046	17.56

Los gráficos 64.1 y 64.2 muestran las relaciones anteriores y posteriores. Inicialmente parece que los puntos no tienen relación alguna entre sí, pero después de la corrección hay una correlación significativa entre el contenido químico real y el calculado.

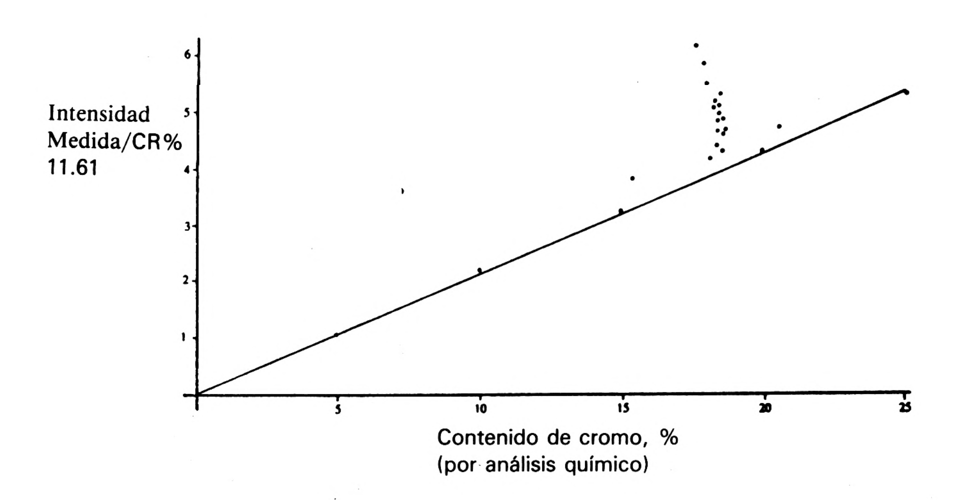


Fig. 64.1. Lecturas de los modelos no corregidas

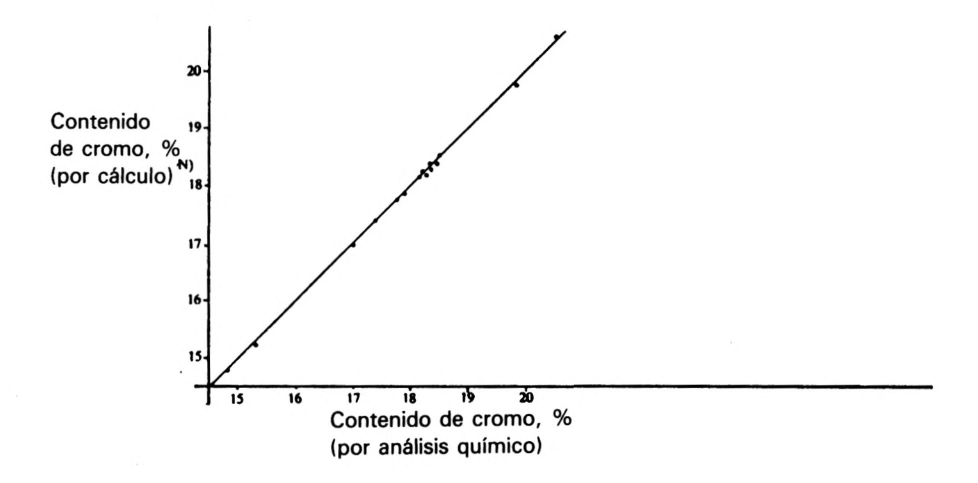


Fig. 64.2. Porcentaje real versus porcentaje calculado de Cr en los modelos

La relación lineal tiene la forma

A*%Cr+B*%Ni+C*%Nb+D*%TI+E*%Mo=(%Cr/I−11.61)

Puede que esto parezca complicado, pero es así porque la aleación más sencilla que puede usted obtener de hierro y cromo da para empezar un gráfico en forma de curva. Puede verificar fácilmente la corrección de la forma de la ecuación con sólo trazar los primeros cinco puntos.

Las 23 ecuaciones representadas por los datos se resolvieron con la rutina de regresión y dieron los siguientes coeficientes:

Elemento	*Coeficiente*
Cromo	0.21578
Niquel	0.05112
Niobio	0.40917
Titanio	0.49890
Molibdeno	0.36674

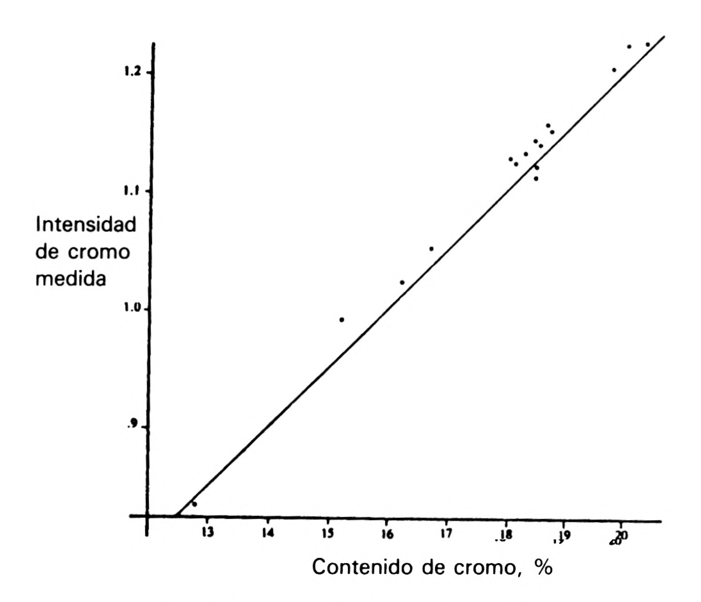


Fig. 64.3. Lecturas no corregidas de muestras de aceros

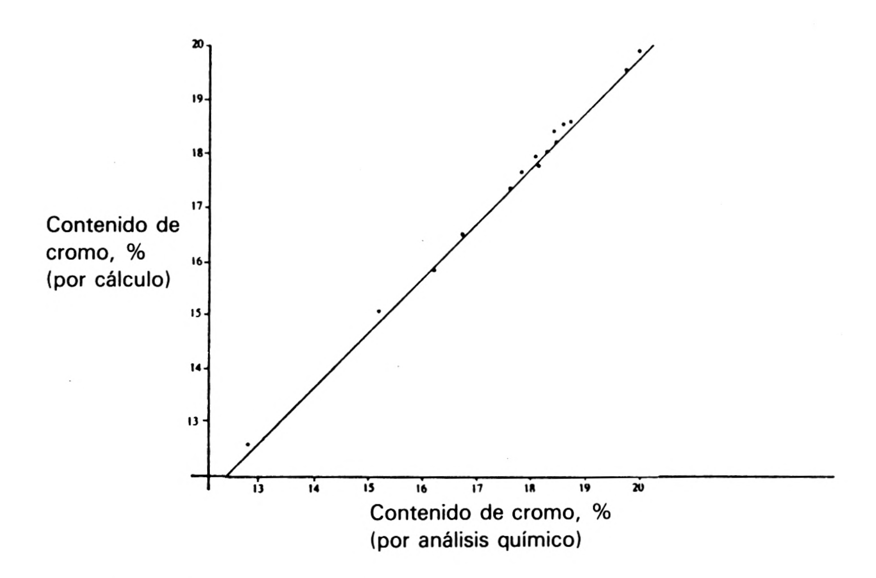


Fig. 64.4. Lecturas de las muestras de aceros una vez corregidas

Sin embargo, la verdadera prueba es tomar otro grupo de aceros sin relación y aplicar los coeficientes a las medidas obtenidas de ellos. Esto se muestra en los gráficos 64.3 y 64.4, y puede observarse que, tras efectuar la corrección, los análisis tienen

una precisión de ±0.15% de Cr. Teniendo en cuenta que hay errores en las medidas de los rayos X y en los análisis químicos, este resultado es satisfactorio.

Sin embargo, vamos a hacerle una advertencia: para obtener coeficientes sensibles debe tener un buen margen de variación en los factores, y nunca debe tratar de calcular las correcciones fuera del margen dentro del cual se obtuvieron en un principio los coeficientes de regresión. Intente comprobar siempre su ecuación con un conjunto distinto de cifras independientes.

65. Rotación de puntos en torno al origen

Esta subrutina toma un grupo de coordenadas x,y y calcula sus nuevas posiciones cuando el grupo rota en torno al origen dentro de un número de posiciones que se encuentran a la misma distancia. Se desarrolló para las máquinas de moler CNC que producen figuras con un eje de simetría, tales como ruedas dentadas o engranajes de muelas. Las coordenadas x,y son los puntos de ruptura donde el perfil cambia de curvatura.

El ángulo de rotación (θ) se calcula a partir del número de sectores. S[2,B] almacena la lista de las coordenadas x,y. R es la matriz de rotación e I es una matriz unitaria que se usa para constituir la matriz de respuesta T.

En el primer ciclo del bucle externo I, las coordenadas originales se almacenan en T. La matriz I se transforma entonces en una matriz de rotación θ y multiplica las coordenadas para obtener la primera posición desplazada. I se convierte entonces en una matriz de rotación 2θ, y así sucesivamente.

Observe el uso de ROUND para restringir el número de posiciones decimales impresas y el de ERASE para permitir que se emplee DIM en el programa y, por tanto, evitar tener siempre a cero todos los valores de las matrices.

Para efectuar una rotación sobre un punto (A,B) en vez de sobre el origen, reste A a la coordenada x y B a la coordenada y durante la rutina INPUT. Después de llevar a cabo la rotación, sume A y B a las coordenadas x e y transformadas, respectivamente. (Véase 'Rotación universal'.)

```
10 REM Rotacion de puntos en torno al origen
20 DEFINT B,J,K,L,N: DEF FN A$(A)=MID$(STR$(A),2)
30 INPUT "No. de sectores";N: IF N<=0 THEN 30
40 INPUT "No. de puntos";B: IF B<=0 THEN 40
50 DIM S[2,B]
60 TETA=2*PI/N
70 CT=COS(TETA): ST=SIN(TETA)
80 CLS: PRINT "Introduzca las coordenadas x e y"
90 FOR P=1 TO B
100 LOCATE 1,P+3: PRINT "x";FN A$(P);"=";: INPUT S[1,P]: LOCATE 20,P+3:
PRINT "y";FN A$(P);"=";: INPUT S[2,P]
```

```
110 NEXT
120 PRINT: PRINT "       Listo? (S/N)";: INPUT K$: K$=UPPER$(K$)
130 IF K$<>"S" THEN ERASE S: CLS: GOTO 30
140 GOSUB 1000: GOSUB 2000: END
1000 DIM R[2,2],I[2,2],T[2,B,N]
1010 R[1,1]=CT: R[2,2]=CT: R[1,2]=-ST: R[2,1]=ST: I[1,1]=1: I[2,2]=1
1020 FOR I=1 TO N: FOR J=1 TO B: FOR K=1 TO 2: FOR L=1 TO 2
1030 T[K,J,I]=T[K,J,I]+I[K,L]*S[L,J]
1040 NEXT L,K,J
1050 FOR J=1 TO 2: DIM Y[2]
1060 FOR K=1 TO 2: FOR L=1 TO 2
1070 Y[K]=Y[K]+R[K,L]*I[L,J]
1080 NEXT L,K
1090 FOR K=1 TO 2: I[K,J]=Y[K]
1100 NEXT K: ERASE Y: NEXT J,I
1110 RETURN
2000 CLS
2010 FOR I=1 TO N: FOR J=1 TO B
2020 PRINT "(X";FN A$(J);"),";FN A$(I);"=";ROUND(T[1,J,I],6)
2030 PRINT "(Y";FN A$(J);"),";FN A$(I);"=";ROUND(T[2,J,I],6)
2040 NEXT J,I
2050 ERASE T,I,R
2060 RETURN
```

Ejemplo

```
No. de sectores? 4
No. de puntos? 1
Introduzca las coordenadas x e y
X1 ? 10          Y1 ? 20
Listo? (S/N)?

(X1),1 = 10
(Y1),1 = 20
(X1),2 = -20
(Y1),2 = 10
(X1),3 = -10
(Y1),3 = -20
(X1),4 = 20
(Y1),4 = -10
```

66-69. Redondeo de números

La operación INT(X) siempre redondea hacia abajo, y la función FIX(X) da la parte entera de un número. Una función del tipo INT(X*1000+.5)/1000 suprimirá un exceso de posiciones decimales (en este caso, redondeándolas a tres), pero hay ocasiones en las que son necesarias otras operaciones de redondeo. ROUND(X,N) permite redondear a N posiciones.

66. Redondeo al entero superior

```
10 REM Redondeo a entero
20 INPUT X
30 GOSUB 1000: GOSUB 2000: END
1000 IF X=INT(X) THEN RETURN
1010 X=1+INT(X): RETURN
2000 PRINT X: RETURN
```

67. Redondeo al entero más próximo

Este programa es equivalente a ROUND(X,O).

```
10 REM Redondeo al entero mas proximo
20 INPUT X
30 GOSUB 1000: GOSUB 2000: END
1000 X=INT(X+0.5): RETURN
2000 PRINT X: RETURN
```

68. Redondeo a N posiciones decimales

Con fines de impresión, habitualmente es posible invocar a la sentencia PRINT USING para especificar el número de posiciones decimales. Sin embargo, puede haber ocasiones en las que sea más apropiada la siguiente subrutina, particularmente cuando usted no sepa el tamaño de la parte entera. ROUND(X,N) es prácticamente idéntica, pero no suministra ceros adicionales.

```
10 REM Redondeo a n decimales
20 INPUT "Numero";A$: INPUT "Numero de decimales";N
30 GOSUB 2000: END
2000 IF N=0 THEN PRINT INT(VAL(A$)+0.5): RETURN
2010 L=LEN(A$)
2020 FOR P=16 TO L: IF MID$(A$,P,1)="." THEN 2040
2030 NEXT: PRINT A$+"."+STRING$(N,48): RETURN
2040 IF N+P>=L THEN PRINT A$+STRING$(N+P-L,48): RETURN
2050 PRINT LEFT$(A$,P+N-1)+MID$(STR$(INT(VAL(MID$(A$,P+N))
/10↑(L-N-P)+0.5)),2)
2060 RETURN
```

69. Redondeo a N cifras significativas

Una buena forma de indicar la precisión es reducir los números a unas cuantas cifras significativas, ya que se puede suponer que el número se aproxima a un dígito en la última posición decimal o a un dígito distinto de cero, si se trata de un número entero.

La subrutina se encarga primero del caso trivial de N=0. A continuación separa los números que contienen un punto decimal de los enteros. Estos últimos necesitan que se los trunque o que se les añada ceros adicionales tras un punto decimal. La línea 2050 trabaja con los números decimales mayores que uno que necesitan un aumento, y después, en SUB 2080, se averigua la posición del punto decimal de los números menores que uno. Los números son truncados o aumentados, según sea necesario. STRING$(N,48) proporciona los ceros.

```
10 REM redondeo a n cifras significativas
20 INPUT "numero"; A$: INPUT "numero de cifras significativas";N
30 GOSUB 2000: END
2000 IF N=0 THEN PRINT "0": RETURN
2010 L=LEN(A$)
2020 FOR P=1 TO L: IF MID$(A$,P,1)="." THEN 2050
2030 NEXT:IF N>=L THEN PRINT A$+"."=STRING$(N-L,48): RETURN
2040 PRINT LEFT$(STR$)INT(VAL)LEFT$(A$,N+1))/10+0.5)),N+1)
+STRING$(L-N,48): RETURN
2050 IF N<=L AND VAL (A$)>1 THEN PRINT A$+STRING$(N-L+1,48):
RETURN
2060 IF VAL(A$)-1 THEN GOSUB 2080: IF N>L-Q+1 THEN PRINT
A$+STRING$(N+Q-L-1,48: RETURN ELSE PRINT INT((VAL(A$))
*10↑(Q+N-P-1)+0.5)/10↑(Q+N-P-1): RETURN
2070 PRINT INT((VAL(A$))*10↑(N-P+1)+0.5)/10↑(N-P+1): RETURN
2080 FOR Q=P+1 TO L: IF MID$(A$,Q,1)<>"0" THEN RETURN
2090 NEXT: RETURN
```

70. Borrado (o llenado)

Esta subrutina borra todo lo que hay dentro de un triángulo, o pinta sobre el papel con el color de la tinta 1 (INK 1). El corazón del programa es muy simple y usa la orden PLOT para sobreimprimir pixels con el color del fondo. Las complicaciones surgen al cubrir todas las orientaciones del triángulo, incluidas aquéllas con lados horizontales y verticales.

En la línea 1000 se vuelve a etiquetar las esquinas, de forma que 1 es el punto superior en la pantalla y 3 el inferior. Se ha establecido un bucle para realizar la trayectoria entre Y1 e Y3, y, atravesando el triángulo, se ha trazado un compartimento horizontal de un pixel de profundidad y del color del fondo. Antes del comienzo del bucle es necesario averiguar las pendientes de los lados sin provocar la división por cero, y después definir P12, P23 y P31 (los recíprocos de las pendientes).

T determina si PLOT va de R a L o de L a R, e INT(1+ABS(P31)) etc. sirve para detener el borrado de las líneas del triángulo. El triángulo se puede suprimir, si es necesario, cambiando el signo de estos términos u omitiéndolos.

Todos los perfiles de polígonos están formados por triángulos, y así la subrutina se puede extender a otras figuras.

```
10 REM Borrado o llenado
20 BORDER 15: INK 0,5: INK 1,20: INK 2,24
30 A$="Este es un programa erradicador que borra los pixels del interior de un triángu
lo utilizando la función PLOT para pintar sobre el papel con el color de tinta 1, INK1"
40 CLS: PEN 2: PAPER 0: PRINT " Escriba las coordenadas de los extremos del
triángulo"
50 LOCATE 1,4: PRINT "X1=";: INPUT X1: LOCATE 20,4: PRINT "Y1=";:
INPUT Y1
60 LOCATE 1,8: PRINT "X2=";: INPUT X2: LOCATE 20,8: PRINT "Y2=";:
INPUT Y2
70 LOCATE 1,12: PRINT "X3=";: INPUT X3: LOCATE 20,12: PRINT "Y3=";:
INPUT Y3
80 CLS: GOSUB 1000: END
1000 IF Y2>Y3 THEN SW=X2: X2=X3: X3=SW: SW=Y2: Y2=Y3: Y3=SW
1010 IF Y1>Y2 THEN SW=X1: X1=X2: X2=SW: SW=Y1: Y1=Y2: Y2=SW:
GOTO 1000
1020 FOR P=1 TO 7: PRINT A$: NEXT
1030 PLOT X1,Y1: DRAW X2,Y2: DRAW X3,Y3: DRAW X1,Y1
1040 IF X1<>X2 THEN M12=(Y1-Y2)/(X1-X2): GOTO 1060
1050 P12=0: GOTO 1070
1060 IF Y1<>Y2 THEN P12=1/M12
1070 IF X2<>X3 THEN M23=(Y2-Y3)/(X2-X3): GOTO 1090
1080 P23=0: GOTO 1100
1090 IF Y2<>Y3 THEN P23=1/M23
1100 IF X3<>X1 THEN M31=(Y3-Y1)/(X3-X1): GOTO 1120
1110 P31=0: GOTO 1130
1120 IF Y3<>Y1 THEN P31=1/M31
1130 IF Y1=Y2 THEN 1190 ELSE T=SGN(P12-P31)
1140 FOR Y=Y1+1 TO Y2+(Y2=Y3)*2
1150 X=X1-Y1*P31+Y*P31+T*INT(1+ABS(P31))
1160 Z=T*(ABS((Y1-Y)*(P31-P12))-T*INT(1+ABS(P12))+T*INT(1+ABS
(P31)))
1170 PLOT X,Y: DRAWR Z,0
1180 NEXT Y
1190 IF Y2=Y3 THEN RETURN ELSE T=SGN(P31-P23)
1200 FOR Y=Y2+1-(Y1=Y3)*2 TO Y3-1
1210 X=X1-Y1*P31+Y*P31+T*INT(1+ABS(P31))
1220 Z=T*(ABS(X2-X1+Y*(P23-P31)+Y1*P31-Y2*P23)+T*INT(1+ABS
(P31))-T*INT(1+ABS(P23)))
1230 PLOT X,Y: DRAWR Z,0
1240 NEXT: RETURN
```

71. Ahorro de memoria

El Amstrad 464 tiene 43903 bytes disponibles para programas; esta cantidad debería ser suficiente para la mayoría de los objetivos. No obstante, a causa del desarrollo de sus habilidades en BASIC y código de máquina, puede que usted escriba programas en los que la memoria sea algo precioso. Sin embargo, a través de unos pasos relativamente sencillos, puede ahorrar fácilmente cantidades significativas de memoria.

Conviene grabar el tamaño de los programas principales; si son demasiados grandes, trate de reducirlos. Para este fin es útil PRINT HIMEM−FRE(0), que se puede añadir al final del programa.

Aparte del problema de legibilidad, no es útil usar nombres largos, ya que se debe interpretar cada letra. Una letra, con o sin dígito, da 268 variables, y dos letras 667 (evitando IF, FN, ON, PI, etc.), lo cual debería ser suficiente. El uso sensato de X, X% y X$ como tres variables distintas puede combinar el nombre de dos símbolos con el tipo correcto.

Sin embargo, en general es mejor usar DEFINT, DEFREAL y DEFSTR al principio del programa, y escoger, por ejemplo, A-H como cadenas, I-T como enteros y U-Z como números reales. En determinados casos, la definición puede estar supeditada a una sentencia. Por ejemplo, DEFINT A: DIM A![N,M] asegura que A1, A2, etc. son enteros, pero la matriz A![N,M] aún puede contener números reales.

El uso de DEF FN establece si la misma expresión tiene que usarse más de una vez en un programa, aunque la cantidad de espacio ahorrado depende de la complejidad de la expresión. Le advertimos, sin embargo, que en Amstrad BASIC no es posible usar una función definida como argumento de una sentencia DIM o como una variable de una matriz. Por ejemplo,

```
DEF FN A(A)=A*A
DIM X[FN A(4)]
```

no funciona, y el uso de FN en una variable de una matriz desordena los bucles FOR...NEXT, las sentencias GOSUB...RETURN y las WHILE...WEND, alterando el puntero de pila. Esto tiene como resultado la aparción de los mensajes Unexpected NEXT, RETURN o WEND (NEXT, RETURN o WEND no esperado).

Por tanto, no use sentencias como

```
Z=X[FN A(2)]
```

En vez de esto, introduzca otra variable

```
A=FN A(2): Z=X[A]
```

El uso de GOSUB ahorra espacio si la rutina se usa más de una vez, pero el empleo de ELSE con IF...THEN evita a menudo tener que usar GOSUB.

Las matrices emplean gran cantidad de memoria. Por ejemplo, en el apartado 'Cristal cúbico', I(3,8,6) usa 590 enteros y 1250 números reales, pero, afortunadamente,

el espacio se puede volver a usar más tarde en el programa, con la ayuda de ERASE, una vez que no se necesitan los datos.

En Amstrad, usted no puede almacenar variables en sentencias DATA, de forma que debe especificar el tipo correcto en la instrucción READ. Véase el apartado 'Almacenamiento de cadenas' para encontrar los métodos más eficaces de almacenamiento de datos.

Siempre que sea posible, vuelva a utilizar los nombres de las variables, por ejemplo para bucles no anidados. Este ejemplo

```
FOR A=1 TO 10: PRINT A: NEXT
FOR B=1 TO 20: PRINT B: NEXT
```

usa nueve bytes más que

```
FOR A=1 TO 10: PRINT A: NEXT
FOR A=1 TO 20: PRINT A: NEXT
```

porque se almacena el último valor de B así como el de A.

Las líneas de sentencias múltiples economizan memoria y, siempre que no sean demasiado largas, no dificultan la legibilidad.

Las sentencias REM son muy valiosas para la comprensión de un programa, pero es mejor colocarlas en la versión impresa del programa que en la residente en la memoria. En los programas del libro, las sentencias REM están a menudo en las líneas que acaban en 9, de forma que pueden ser ignoradas con la facilidad AUTO. Los GOTO y GOSUB van a las líneas posteriores a la que contiene a REM; dichas líneas deben ser un múltiplo de diez.

72. Desplazamiento

Esta subrutina debería incluirse en los programas que generan gran cantidad de datos escritos en la pantalla, de forma que se pueda evitar el procedimiento, un tanto accidental, de detención y comienzo con <ESC>.

La rutina emplea VPOS(#0) para detectar la posición de impresión actual y se detiene cuando llega a 20. Una vez borrada la pantalla, se puede restablecer la impresión cuando se quiera.

```
10 REM Desplazamiento
20 INPUT "Numero de valores";N: REM N>25 para que sea relevante
30 DIM A[N]
40 FOR P=1 TO N: A[P]=P: NEXT: REM Por ejemplo
50 GOSUB 2000: END
2000 CLS: FOR P=1 TO N: PRINT "P=";P,"P↑2=";A[P]*A[P]: REM Por ejemplo
2010 IF VPOS(#0)>20 THEN GOSUB 2030: CLS
2020 NEXT: RETURN
2030 LOCATE 1,22: PRINT "Para continuar pulse cualquier tecla"
2040 IF INKEY$="" THEN 2040
2050 RETURN
```

73-76. Progresiones

Habitualmente, se puede escribir las progresiones como operaciones de bucle, que resultan más fáciles de calcular que una serie de términos individuales. Por tanto, las progresiones exponenciales pueden aparecer así

$$S=1+\frac{x}{2}(1+\frac{x}{3}(1+\frac{x}{4}(1+x(1+ \text{ etc.}$$

las progresiones geométricas así

$$S=a(1+r(1+r(1+r(1+ \text{ etc.}$$

las progresiones aritméticas así

$$S=n*a+d(1+2+3+4+ \text{ etc.}$$

y las progresiones binomiales así

$$S=1+\frac{nx}{1}(1+\frac{(n-1)}{2}x(1+\frac{(n-2)}{3}x(1+ \text{ etc.}$$

La forma del bucle empleado se hace obvia estudiando las ecuaciones anteriores.

Este grupo de subrutinas calcula la suma de N términos de varias progresiones diferentes. (Se emplea para estudiar la convergencia y el cálculo de errores provocados cuando se ignora los términos de orden superior.)

73. Progresiones exponenciales

$$S=1+\frac{x}{1}(1+\frac{x}{2}(1+\frac{x}{3}(1+\frac{x}{4}(1+\ldots$$

```
10 REM Progresiones exponenciales
20 INPUT "X=";X: INPUT "N=";N
30 GOSUB 1000: GOSUB 2000: END
1000 S=0: E=1
1010 FOR P=1 TO N
1020 S=S+E: E=E*X/P: NEXT
1030 RETURN
2000 PRINT TAB(8);"S=";S
2010 RETURN
```

74. Progresiones geométricas

$$S=a+ar+ar^2+ar^3+ar^4+\ldots \qquad (=a(r^n-1)/(r-1))$$

```
10 REM Progresiones geometricas
20 INPUT "A=";A
30 INPUT "R=";R
40 INPUT "N=";N
50 GOSUB 1000: GOSUB 2000: END
1000 S=0: G=A
1010 FOR P=1 TO N
1020 S=S+G:G=G*R: NEXT
1030 RETURN
2000 PRINT TAB(8);"S=";S
2010 RETURN
```

75. Progresiones aritméticas

$$S=a+(a+d)+(a+2d)+(a+3d)+\ldots$$

```
10 REM Progresiones aritmeticas
20 INPUT "A=";A
30 INPUT "D=";d
40 INPUT "N=";N
50 GOSUB 1000: GOSUB 2000: END
1000 S=0
1010 FOR P=1 TO N
1020 S=S+A+(P-1)*D: NEXT
1030 RETURN
2000 PRINT TAB(8);"S=";S
2010 RETURN
```

76. Progresiones binomiales

$$S=1+nx+n(n-1)x^2/2!+n(n-1)(n-2)x^3/3!+\ldots$$

$$=(1+x)n \text{ (si } |x|<1 \text{ cuando } n<>\text{entero positivo)}$$

```
10 REM Progresiones binomiales
20 INPUT "X=";X
30 INPUT "N=";N
40 INPUT "Numero de terminos";R
50 GOSUB 1000: GOSUB 2000: END
1000 S=0: Z=1: M=N
```

```
1010 FOR P=1 TO R
1020 S=S+Z: Z=Z*M*X/P: M=M-1: NEXT
1030 RETURN
2000 PRINT TAB(8);"S=";S
2010 RETURN
```

77. Impresión lateral

Esta subrutina permite que los números sean impresos de forma que lo que sobra por la derecha se escriba debajo del número, y no en la parte opuesta de la pantalla.

Emplea dos operaciones de incremento, es decir, FOR P=1 TO LEN(N$) y C=C+1. En el límite de la pantalla, C retrocede a C=W+P−LEN(N$) para escribir el resto de los números en la línea de abajo.

```
10 REM Impresion lateral
20 INPUT "MODE";M: IF M<0 OR M>2 THEN 20
30 MODE M: W=21-20*(M=1 OR M=2)-40*(M=2)
40 INPUT "Numero";N
50 INPUT "Fila";F: REM 1<=F<=23
60 INPUT "Columna";C: REM 1<=C<=W-1
70 GOSUB 2000: END
2000 N$=STR$(N): L=LEN(N$)
2010 CLS: FOR P=1 TO L
2020 LOCATE C,F: PRINT MID$(N$,P,1)
2030 C=C+1
2040 IF C=W THEN F=F+1: C=W+P-L
2050 NEXT
2060 LOCATE 1,1
2070 RETURN
```

78. Sistemas de ecuaciones

Esta solución emplea el método de la matriz inversa, por el que si las ecuaciones, por ejemplo

$$a_1*X+b_1*Y+c_1*Z=K_1$$
$$a_2*X+b_2*Y+c_2*Z=K_2$$
$$a_3*X+b_3*Y+c_3*Z=K_3$$

están escritas en forma de matrices, es decir

$$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} K_1 \\ K_2 \\ K_3 \end{bmatrix}$$

entonces

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}^{-1} * \begin{bmatrix} K_1 \\ K_2 \\ K_3 \end{bmatrix}$$

La inversa se obtiene por el método de Gauss-Jordan y se multiplica por la matriz de constantes.

SUB 1060 comprueba si hay ceros en la diagonal y suma dos líneas en SUB 1080 para evitar un fallo en la línea 1110. La línea siguiente busca un cero en el determinante 2×2 superior izquierdo y, si lo encuentra, suma la última ecuación a la primera. SUB 1110 es la rutina de inversión de matrices que también comprueba, en la línea 1190, que el determinante no es cero (una condición previa para hallar la inversa). De vuelta de SUB 1110, la inversa se multiplica por la matriz de constantes E[K,N+1] y se imprime S[J] empleando CHR$(90−N+J) para escribir los nombres apropiados de las variables x, y, z, etc.

El método no se limita a las ecuaciones de 8×8, sino que la limitación surge de la rutina de introducción de datos y puede ser solventada fácilmente.

Esta subrutina comprueba si hay solución y, si es así, resuelve las ecuaciones.

```
10 REM Sistema de ecuaciones
20 DEFINT I,J,K,N: DEF FN A$(A)=MID$(STR$(A),2)
30 INPUT "Numero de variables";N: IF N<=1 THEN 20
40 CLS: PRINT "Introduzca los coeficientes"
50 FOR J=1 TO N: FOR K=1 TO N+1
60 PRINT "E("+FN A$(J)+",";FN A$(K)+")=";: INPUT E(J,K)
70 NEXT K,J
80 REM Alternativamente, use la rutina de introduccion de datos (ecuaciones
lineales)
90 GOSUB 1000: GOSUB 2000: END
1000 DIM S[N]: GOSUB 1060: IF D=0 THEN RETURN
1010 IF N<1 THEN IF E[1,1]*E[2,2]=E[1,2]*E[2,1] THEN GOSUB 1100
1020 GOSUB 1110: IF D=0 THEN RETURN
1030 FOR J=1 TO N: FOR K=1 TO N
1040 S[J]=S[J]+E[J,K]*E[K,N+1]
1050 NEXT K,J: RETURN
1060 D=1: FOR I=1 TO N: IF E[I,I]=0 THEN GOSUB 1080
1070 NEXT: RETURN
1080 FOR J=1 TO N
1090 NEXT: D=0: RETURN
1100 FOR I=1 TO N+1: E[1,I]=E[1,I]+E[N,I]: NEXT: RETURN
1110 FOR I=1 TO N: IF J=I THEN 1190
1130 E[J,I]=E[J,I]*E[I,I]
1140 FOR K=1 TO N: IF K=I THEN 1180
```

```
1150 E[J,K]=E[J,K]-E[J,I]*E[I,K]
1160 IF J<>N+(I=N) THEN 1180
1170 E[I,K]=-E[I,I]*E[I,K]
1180 NEXT
1190 NEXT: IF I<>N THEN IF ABS(E[I+1,I+1])<0.0000001 THEN D=0: RETURN
1200 NEXT: RETURN
2000 CLS: IF D=0 THEN PRINT "No hay solucion.": ERASE S: RETURN
2010 LOCATE 7,1: PRINT "La solucion es": PRINT
2020 FOR J=1 TO N: PRINT TAB(3);CHR$(90-N+J);"=";S[J]: PRINT
2030 NEXT
2040 ERASE S: RETURN
```

Ejemplo

```
N=6         Sus ecuaciones son

U/Y     V/Z     W       X           Const.

a1*u  + b1*v  + c1*w  + d1*x    +
e1*y  + f1*z                        =k1
a2*u  + b2*v  + c2*w  + d2*x    +
e2*y  + f2*z                        =k2
a3*u  + b3*v  + c3*w  + d3*x    +
e3*y  + f3*z                        =k3
a4*u  + b4*v  + c4*w  + d4*x    +
e4*y  + f4*z                        =k4
a5*u  + b5*v  + c5*w  + d5*x    +
e5*y  + f5*z                        =k5
a6*u  + b6*v  + c6*w  + d6*x    +
e6*y  + f6*z                        =k6

Introduzca los valores

U/Y     V/Z     V       X           Const.

7     + 5     + -4    + 2       +
13    + 6                           = 15
4.5   + 2     + 19    + 24      +
-7    + 9.6                         = 3
5.5   + 13    + 9.43  + -5.46   +
5     + 32                          = 5.44
-7.9  + 6.1   + 3.2   + 2.7     +
5     + 8                           = 23
6.9   + 5.3   + 6.7   + 2       +
0     + 5.7                         = 4.8
-5.4  + 1     + 6.8   + 4       +
3.88  + 4                           = 87
```

La solucion es

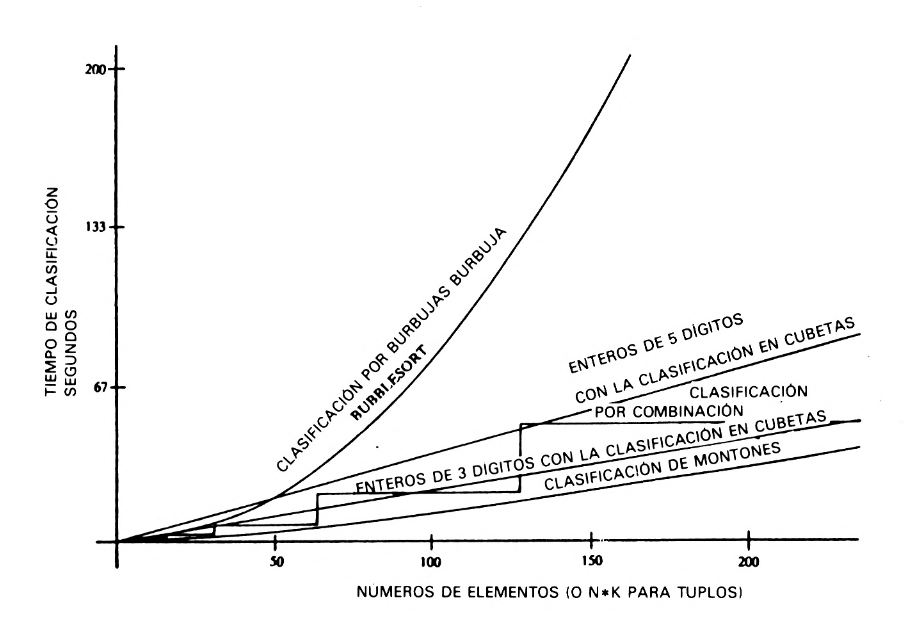
u= −1.43955734
v= −7.19042574
w=   10.7642367
x= −3.94752449
y=   9.53563634
z= −1.997904118

(Duracion del calculo 5 segundos)

79-84. Clasificación

La clasificación de palabras o números es una operación muy corriente y hay muchos algoritmos capaz de llevarla a cabo. Sin embargo, la elección del mejor de ellos necesita algunas ideas; aquí damos ejemplos de cuatro algoritmos distintos que ilustran diferentes elecciones. La consideración principal es la velocidad del algoritmo y, por tanto, cuánto aumenta el tiempo según el número de elementos que hay que ordenar.

En los programas se usa habitualmente la clasificación por burbujas, pero en la práctica ésta es muy pobre cuando el número de elementos es muy grande, ya que el tiempo se incrementa al cuadrado del número. La clasificación de montones es mucho más eficaz para números grandes, ya que el tiempo sólo aumenta como *n log n*. Las razones de esto se pueden ver comparando el funcionamiento de los algoritmos de clasificación. Si tenemos una lista de elementos ordenados, excepto el elemento mayor, que está en el extremo equivocado, con la clasificación por burbujas dicho elemento se desplaza a la posición correcta saltando los lugares de uno en uno, mientras que con la clasificación de montones y la clasificación por combinación se desplaza una posición la primera vez, después dos posiciones, a continuación cuatro posiciones, ocho, dieciséis, etc., y así se llega al lugar correcto con mucha más rapidez.



La clasificación por combinación es casi tan veloz como clasificación de montones, pero, al necesitar 2^n valores en el algoritmo, hay que añadir valores adicionales, y lleva más tiempo ordenar 65 valores que 128. Sin embargo, la primera clasificación se puede realizar durante la entrada (INPUT).

La clasificación en cubetas se ha usado durante mucho tiempo en las máquinas de clasificación de tarjetas perforadas, y es útil para ordenar números enteros o cadenas. Una lista de n enteros o cadenas que contiene k caracteres se puede ordenar en un tiempo proporcional a K por n.

El gráfico muestra cómo funcionan las cuatro rutinas. Evidentemente, aunque la clasificación por burbujas es, con mucho, la rutina más sencilla, no se puede usar con números elevados de elementos.

Hay otras dos rutinas, una para palabras y la otra para ángulos, que implican otras consideraciones distintas del mero hecho de ordenar, es decir, organizan los datos de forma que sean apropiados para un algoritmo de ordenamiento estándar.

En ambos casos, en los programas se ha empleado la clasificación por burbujas, pero cuando haya un número mayor de elementos se debe considerar un algoritmo más eficaz.

79. Clasificación angular

Esta subrutina coloca una lista de coordenadas en orden angular con relación a su centro.

Las primeras ocho líneas encuentran los valores superior e inferior de x e y y se elige el punto (XM,YM) a mitad de camino entre ellos. Se calcula y se examina los ángulos entre la horizontal y las líneas que unen cada punto con ese punto central para comprobar en qué cuadrante están. Esto se debe a que la tangente es positiva en el primer y tercer cuadrante y negativa en los otros dos. Si es necesario, se ajusta el ángulo sumando PI o 2*PI, y luego se lo almacena en B[N]. Para ordenar los ángulos se emplea una rutina de clasificación por burbujas, 1140, y las coordenadas siguen por simpatía. Por lo que se refiere a la imagen, el 0.92 tiene el fin de corregir la relación de aspecto de la pantalla.

Ejemplo

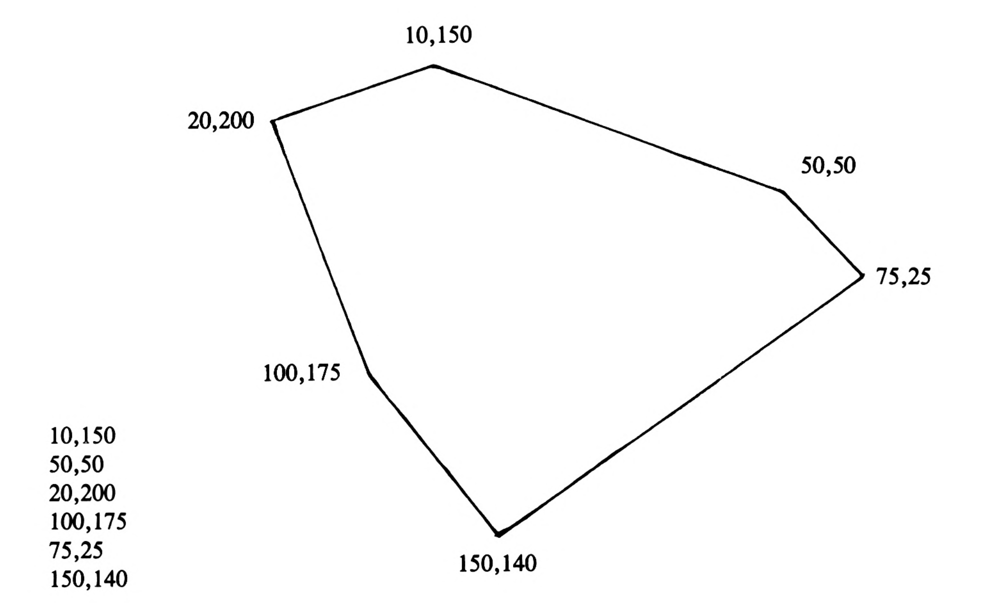


Fig. 79.1. Clasificación angular.

```
10 REM Clasificacion en angulo
20 DEF FN A$(A)=MID$(STR$(A),2)
30 INPUT "Numero de pares de lecturas";N: IF N<=0 THEN 30
40 DIM A[1,N]: CLS
50 FOR P=1 TO N: LOCATE 5,P+2: PRINT "x";FN A$(P);"=";: INPUT A[0,P]
60 LOCATE 21,2+P: PRINT "y";FN A$(P);"=";: INPUT A[1,P]: NEXT
70 GOSUB 1000: GOSUB 2000: END
1000 XX=0: YX=0: XN=A[0,1]: YN=A[1,1]
1010 FOR P=1 TO N
1020 IF A[0,P]>=XX THEN XX=A[0,P]
1030 IF A[1,P]>=YX THEN YX=A[1,P]
1040 IF A[0,P]<XN THEN XN=A[0,P]
1050 IF A[1,P]<YN THEN YN=A[1,P]
1060 NEXT
1070 XM=(XX+XY)/2: YM=(YX+YN)/2
1080 DIM B[N]: FOR P=1 TO N
1090 Z=ATN(A[1,P]-YM)/(A[0,P]-XM)
1100 IF A[0,P]>=XM AND A[1,P]>=YM THEN B[P]=Z
1110 IF A[0,P]<XM AND A[1,P]>=YM OR A[0,P]<XM AND A[1,P]<YM THEN
B[P]=Z+PI
1120 IF A[0,P]>=XM AND A[1,P]<YM THEN B[P]=Z+2*PI
1130 NEXT
1140 Q=0: FOR P=1 TO N-1
```

```
1150 IF B[P+1]<B[P] THEN SW=B[P]: B[P]=B[P+1]: B[P+1]=SW:
SW=A[0,P]: A[0,P]=A[0,P+1]: A[0,P+1]=SW: SW=A[1,P]: A[1,P]=A[1,P+1]:
A[1,P+1]=SW: Q=Q+1
1160 NEXT: IF Q<>0 THEN 1140
1170 RETURN
2000 CLS: TAG
2010 FOR P=0 TO N-1
2020 Z1=1+P: Z2=1+(P+1)MOD N
2030 PLOT A[0,Z1],0.92*A[1,Z1]
2040 DRAW A[0,Z2],0.92*A[1,Z2]: MOVER 5,-5: PRINT FN A$(A[0,Z2]);",";
FN A$(A[1,Z2]);
2050 NEXT
2060 ERASE B: TAGOFF:
2070 RETURN
```

80. Clasificación por burbujas

Esta conocida rutina de clasificación se llama 'por burbujas' porque los números altos ascienden por la lista como si fueran las burbujas de un líquido. *Q* cuenta el número de cambios llevados a cabo. La línea 1010 decide si es necesario un intercambio entre números adyacentes y, si es así, lo realiza. La rutina prosigue el reciclaje hasta que ya no se necesita más intercambios, es decir, hasta que *Q*=0.

Esta subrutina coloca los números en orden descendente. Para cambiar a un orden ascendente sustituya > por < en la línea 1010.

```
10 REM Clasificacion por burbujas
20 N=0
30 WHILE N<=0: INPUT "Numero de elementos";N: WEND
40 DIM A[N]
50 FOR P=1 TO N: INPUT A[P]: NEXT
60 GOSUB 1000: GOSUB 2000: END
1000 Q=0: FOR P=1 TO N-1
1010 IF A[P+1]>A[P] THEN SW=A[P]: A[P]=A[P+1]: A[P+1]=SW: Q=Q+1
1020 NEXT: IF Q<>0 THEN 1000
1030 RETURN
2000 FOR P=1 TO N: PRINT A[P]: NEXT
2010 RETURN
```

81. Clasificación en cubetas

En la clasificación en cubetas, los tuplos se ordenan en términos del último caracter. Después, una vez reconstituida la cadena, en términos del siguiente-al-último carácter, y así sucesivamente. Por ejemplo, los siguientes seis tuplos, BACD, BBCD, ABCD, BBAC, BAAB, AABC, una vez ordenados dan

A
B BAAB
C BBAC, AABC
D BACD, BBCD, ABCD

Lo que da un nuevo orden de BAAB, BBAC, AABC, BACD, BBCD, ABCD. Después de ser clasificados en orden del siguiente-al-último, esto da

A BAAB, BBAC
B AABC
C BACD, BBCD, ABCD
D

Esto se convierte en BAAB, BBAC, AABC, BACD, BBCD, ABCD.

Las dos clasificaciones siguientes dan

A BAAB, AABC, BACD
B BBAC, BBCD, ABCD
C
D

y

A AABC, ABCD
B BAAB, BACD, BBAC, BBCD
C
D

Así pues, el orden final es AABC, ABCD, BAAB, BACD, BBAC, BBCD.

El bucle P trabaja con los caracteres en sentido contrario. A$="" etc. vacía las cubetas. El bucle Q encuentra la coma que separa los tuplos (véase 'Almacenamiento de cadenas') y la sentencia IF coloca punteros hacia el tuplo apropiado, reuniéndose éstos en Z$ preparados para el ciclo siguiente del bucle P.

La rutina alternativa emplea la misma entrada y salida, pero está estructurada para ordenar números enteros.

Para llevar a cabo una clasificación en cubetas con todo el alfabeto, sería conveniente usar un árbol 2-3 para conseguir rapidez. Un árbol con una altura de 3 tiene 27 hojas que acomodarán a 26 letras del alfabeto y a un espacio.

```
10 REM Clasificacion en cubetas (Tuplos de igual longitud formados por A, B, C y D)
20 DEFINT A,B,K,N,P,Q
30 INPUT "Numero de tuplos";N: IF N<=0 THEN 30
40 INPUT "Numero de caracteres en cada tuplo";K: IF K<=0 THEN 40
50 DIM D$[N,K]: Z$=""
60 FOR P=1 TO N: Z$=Z$+STR$(P)+","
70 PRINT "El tuplo No.";P;"es";: INPUT T$
80 FOR Q=1 TO K: D$[P,Q]=MID$(T$,Q,1): NEXT Q,P
90 GOSUB 1000: GOSUB 2000: END
```

```
1000 FOR P=K TO 1 STEP -1: A=1: B=1: A$=""
1010 FOR Q=1 TO LEN(Z$)
1020 IF MID$(Z$,Q,1)="," THEN A=Q: Z=VAL(MID$(Z$,B+(B=2),A-B-(B=2)))
ELSE 1070
1030 IF D$(Z,P)="A" THEN A$=A$+STR$(Z)+",": GOTO 1070
1040 IF D$(Z,P)="B" THEN B$=B$+STR$(Z)+",": GOTO 1070
1050 IF D$(Z,P)="C" THEN C$=C$+STR$(Z)+",": GOTO 1070
1060 IF D$(Z,P)="D" THEN D$=D$+STR$(Z)+","
1070 B=A+1: NEXT
1080 Z$=A$+B$+C$+D$
1090 NEXT
1100 RETURN
2000 A=1: B=1: FOR P=1 TO LEN(Z$)
2010 IF MID$(Z$,P,1)="," THEN A=P: Z=VAL(MID$(Z$,B+(B=2),A-B-(B=2)))
ELSE 2030
2020 FOR Q=1 TO K: PRINT D$(Z,Q);: NEXT: PRINT
2030 B=A+1: NEXT
2040 RETURN
```

```
999 REM Para los numeros use la siguiente subrutina
1000 FOR P=K TO 1 STEP -1: A=1: B=1: A$="": B$="": C$="": D$="": E$="":
F$"": G$="": H$="": I$="": J$=""
1010 FOR Q=1 TO LEN(Z$)
1020 IF MID$(Z$,Q,1)="," THEN A=Q: Z=VAL(MID$(Z$,B+(B=2),A-B-(B=2))):
T$=D$(Z,P) ELSE 1130
1030 IF T$="0" THEN A$=A$+STR$(Z)+",": GOTO 1130
1040 IF T$="1" THEN B$=B$+STR$(Z)+",": GOTO 1130
1050 IF T$="2" THEN C$=C$+STR$(Z)+",": GOTO 1130
1060 IF T$="3" THEN D$=D$+STR$(Z)+",": GOTO 1130
1070 IF T$="4" THEN E$=E$+STR$(Z)+",": GOTO 1130
1080 IF T$="5" THEN F$=F$+STR$(Z)+",": GOTO 1130
1090 IF T$="6" THEN G$=G$+STR$(Z)+",": GOTO 1130
1100 IF T$="7" THEN H$=H$+STR$(Z)+",": GOTO 1130
1110 IF T$="8" THEN I$=I$+STR$(Z)+",": GOTO 1130
1120 IF T$="9" THEN J$=J$+STR$(Z)+","
1130 B=A+1: NEXT
1140 Z$=A$+B$+C$+D$+E$+F$+G$+H$+I$+J$: NEXT
1150 RETURN
```

```
10 REM Clasificacion en cubetas del alfabeto empleando un arbol 2-3
20 DEFINT A,B,H,I,K,L,N,P,Q,V,Z
30 DIM L$[40],M$[13],T$[27]
40 L$="HBKT CFILORUX": M$="QENWADGJMPSVY"
50 FOR P=1 TO 13: L$[P]=MID$(L$,P,1): M$[P]=MID$(M$,P,1): NEXT
60 L$[14]=" ": FOR P=15 TO 40: L$[P]=CHR$(51+P): NEXT
70 INPUT "Numero de tuplos";n
```

```
80 INPUT "Numero maximo de caracteres por tuplo";K
90 DIM D$[N,K]: Z$=""
100 FOR P=1 TO N: Z$=Z$+STR$(P)+","
110 PRINT "El tuplo No.";P;"es";: INPUT T$
120 L=LEN(T$): IF L>K THEN 110
130 T$=UPPER$(T$)+STRING$(K-L,32)
140 FOR Q=1 TO K: D$(P,Q)=MID$(T$,Q,1): NEXT Q,P
150 GOSUB 1000: GOSUB 2000: END
1000 FOR P=K TO 1 STEP -1: A=1: B=1
1010 FOR I=1 TO 27: T$[I]="": NEXT
1020 FOR Q=1 TO LEN(Z$)
1030 IF MID$(Z$,Q,1)="," THEN A=Q: Z=VAL(MID$(Z$,B+(B=2),A-B-(B=2)))
ELSE 1070
1040 V=1: H=1
1050 WHILE H<4: GOSUB 1110: H=H+1: WEND
1060 V=V-13: T$[V]=T$[V]+STR$(Z)+","
1070 B=A+1: NEXT: Z$=""
1080 FOR I=1 TO 27: IF T$[I]<>"" THEN Z$=Z$+T$[I]
1090 NEXT I,P
1100 RETURN
1110 IF D$[Z,P]<=L$[V] THEN V=3*V-1: RETURN
1120 IF D$[Z,P]<=M$[V] THEN V=3*V: RETURN
1130 V=3*V+1: RETURN
2000 A=1: B=1: FOR P=1 TO LEN(Z$)
2010 IF MID$(Z$,P,1)="," THEN A=P: Z=VAL(MID$(Z$,B+(B+2),A-B-(B=2)))
ELSE 2030
2020 FOR Q=1 TO K: PRINT D$[Z,Q];: NEXT: PRINT
2030 B=A+1: NEXT
2040 RETURN
```

82. Clasificación de montones

'Montón' es el nombre para una estructura de árbol binario donde los elementos están distribuidos de tal manera que uno de ellos asociado a cualquier vértice es mayor o igual que los asociados con sus dos hijos. El nivel de una hoja sólo puede diferir un nivel como máximo de cualquier otra hoja.

El diagrama muestra un montón típico. Si hay que almacenar los elementos A1, A2, A3, etc., esto se hace en la matriz A[N]. Las propiedades del montón implican que

```
        A[I]>=A[2*I]        para 1<=I<=N/2
y       A[I]>=A[2*I+1]      para 1<=I<=N/2
```

La primera parte de la rutina es 'amontonar', es decir, establecer el orden en la matriz para darle la propiedad de montón.

Después, la segunda parte de la rutina ordena los elementos de la forma siguiente:

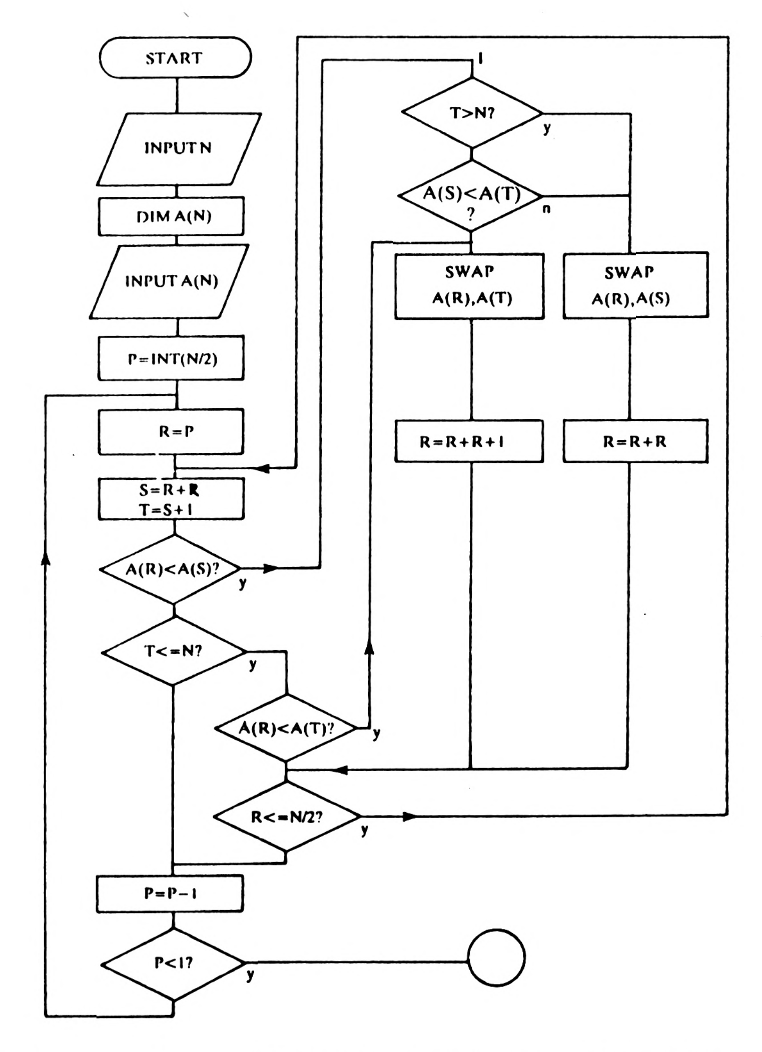
1. Elimina el elemento mayor (que está en la raíz) y lo cambia por el último. Este elemento ya no sigue interviniendo.

2. Reforma los elementos que quedan usando el amontonamiento.

3. Esto lo repite hasta que la matriz contiene todos los elementos colocados en orden ascendente.

Este proceso queda ilustrado en la secuencia de matrices. Los organigramas muestran cómo funciona la subrutina.

```
10 REM Clasificacion de montones
20 INPUT "Numero de valores";M: IF M<=1 THEN 20
30 DIM D[M]
40 FOR P=1 TO M: PRINT "Valor";P;"es";: INPUT D[P]: NEXT
50 GOSUB 1000: GOSUB 2000: END
1000 FOR P=INT(M/2) TO 1 STEP -1: R=P
1010 S=R+R: T=S+1
1020 IF D[R]<D[S] THEN GOSUB 1170: GOTO 1050
1030 IF T<=M THEN IF D[R]<D[T] THEN GOSUB 1200: GOTO 1050
1040 GOTO 1060
1050 IF R<=M/2 THEN 1010
1060 NEXT P
1070 FOR P=M TO 3 STEP -1: SW=D[1]: D[1]=D[P]: D[P]=SW: R=1
1080 S=R+R: T=S+1
1090 IF D[R]<D[S] THEN GOSUB 1150: GOTO 1120
1100 IF T<P THEN IF D[R]<D[T] THEN GOSUB 1200: GOTO 1120
1110 GOTO 1130
1120 IF R<=(P-1)/2 THEN 1080
1130 NEXT P
1140 SW=D[1]: D[1]=D[2]: D[2]=SW: RETURN
1150 IF T>=P THEN 1190
1160 GOTO 1180
1170 IF T>M THEN 1190
1180 IF D[S]<D[T] THEN 1200
1190 SW=D[R]: D[R]=D[S]: D[S]=SW: R=R+R: RETURN
1200 SW=D[R]: D[R]=D[T]: D[T]=SW: R=R+R+1: RETURN
2000 FOR P=1 TO M: PRINT D[P]: NEXT
2010 RETURN
```



N.B. En el programa se usa DIM D(N); en el diagrama DIM A(N)

Fig. 82.1. Amontonamiento.

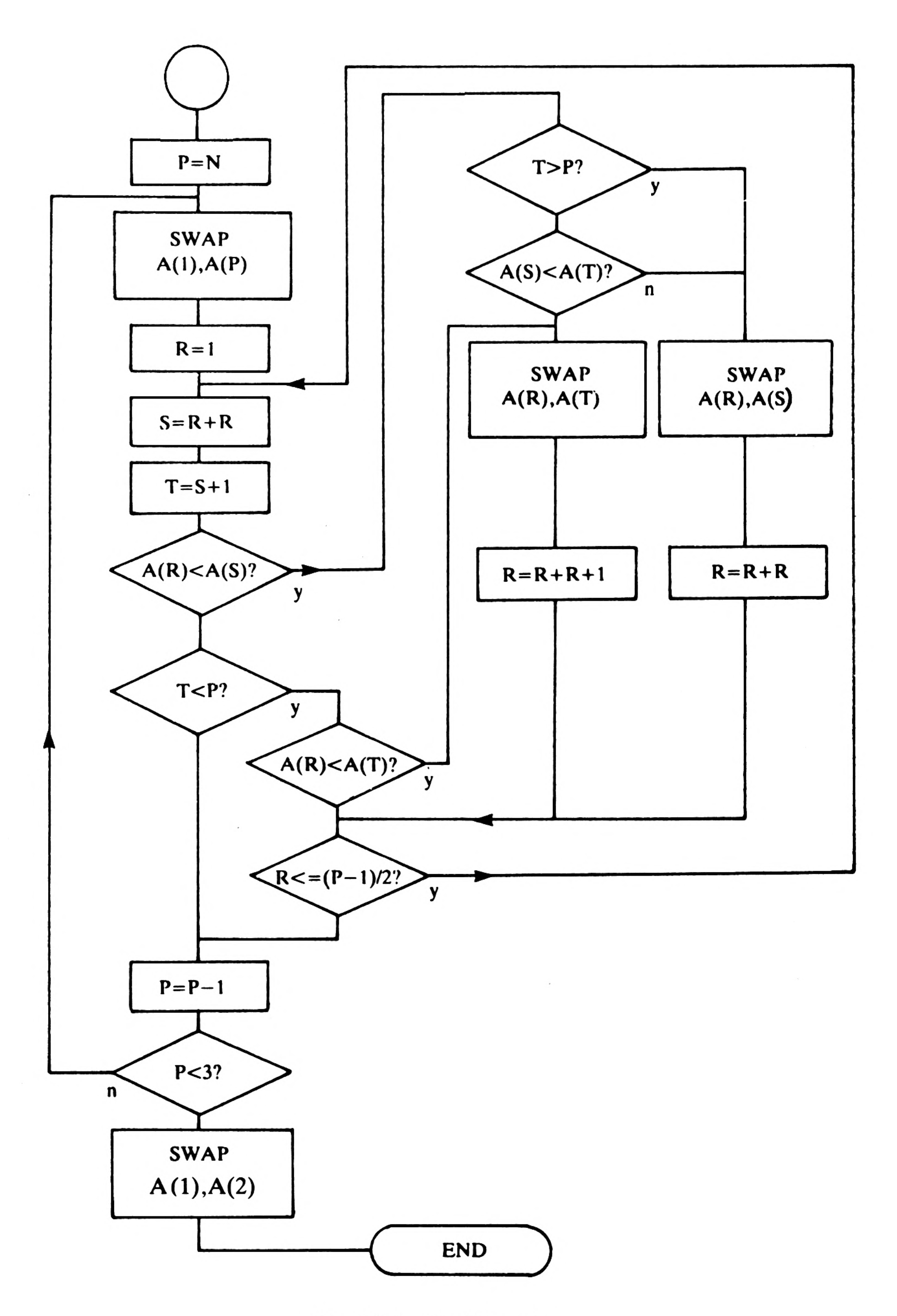


Fig. 82.2. Clasificación.

83. Clasificación por combinación

La clasificación por combinación es una subrutina que ordena una lista de 2n elementos. Si el número de elementos no es una potencia de dos, hay que añadir valores más altos que el mayor de la lista y descartarlos al final. La rutina funciona ordenando primero pares de valores, después combina los pares adyacentes, luego los grupos de cuatro elementos, a continuación los grupos de 8, de 16, etc., empleando la subrutina de combinación. Por ejemplo, la lista

	8,6,4,1,5,6,3,1
se convierte en	6 8,1 4,5 6,1 3
que se convierte en	1 4 6 8,1 3 5 6
que se convierte en	1 1 3 4 5 6 6 8

DEF FN le da formato empaquetado a la entrada (INPUT). Mientras dura ésta, se busca el valor máximo, MX, y se ordena los pares de valores (en la línea 1050). SUB 1200 completa el resto de la matriz A con valores de MX+1. La línea 1080 empieza con los pares de valores y los combina con SUB 1090; a continuación hace lo mismo con los grupos de 4, 8, 16, etc. Como los valores adicionales MX+1 terminan al comienzo de la matriz, la sentencia PRINT, que acaba en 1+S−N, los ignora. Para invertir el orden, establezca el bucle 1+S−N TO S.

```
10 REM Clasificacion por combinacion
20 DEFINT A-C,N,P,X-Z: DEF FN A$(A)=MID$(STR$(A),2)
30 INPUT "Numero de valores";N: IF N<=0 THEN 30
40 M=N: P=0: MX=0
50 WHILE M>1: M=M/2: P=P+1: WEND
60 S=2↑P: DIM A[S],X[S/2+1],Y[S/2+1],Z[S+2]
70 GOSUB 1000: GOSUB 2000: END
1000 FOR A=1 TO N: PRINT "A("+ FN A$(A)+")=";: INPUT A[A]
1010 IF A[A]>MX THEN MX=A[A]
1020 A=A+1: IF A>N THEN GOSUB 1200: GOTO 1070
1030 PRINT "A("+FN A$(A)+")=";: INPUT A[A]
1040 IF A[A]>MX THEN MX=A[A]
1050 IF A[A]>A[A-1] THEN SW=A[A]: A[A]=A[A-1]: A[A-1]=SW
1060 NEXT A: GOSUB 1200: GOTO 1080
1070 IF A[N+1]>A[N] THEN SW=A[N]: A[N]=A[N+1]: A[N+1]=SW
1080 M=2: X[1]=MX+2: Y[1]=X[1]
1090 FOR B=0 TO S/M/2-1: T=2*B*M-1: FOR C=2 TO M+1
1100 X[C]=A[C+T]: Y[C]=A[C+T+M]: NEXT: GOSUB 1220
1120 NEXT: M=M+M: IF M<S GOTO 1090 ELSE RETURN
1200 IF M<>1 THEN FOR A=N+1 TO S: A[A]=MX+1: NEXT
1210 RETURN
1220 X=M+1: Y=X
1230 IF X[X]<Y[Y] THEN Z[X+Y]=X[X]: IF X>1 THEN X=X-1: GOTO 1230
```

```
1240 Z[X+Y]=Y[Y]: IF Y>1 THEN Y=Y-1: GOTO 1230
1250 FOR J=3 TO M+M+2: A[J+T-1]=Z[J]: NEXT: RETURN
2000 FOR A=S TO 1+S-N STEP -1
2010 PRINT A[A]: NEXT
2020 ERASE A,X,Y,Z: RETURN
```

Este programa es para números enteros. Para usarlo con números reales cambie la línea 20 de la forma siguiente:

```
20 DEFINT B,C,N,P: DEF FN A$ etc.
```

84. Clasificación de palabras

Una rutina de clasificación empleada con caracteres ASCII funcionará ordenando dichos caracteres según su código. Para obtener un orden alfabético adecuado hay que convertir todas las letras mayúsculas en minúsculas o viceversa. Los demás símbolos no son importantes para el orden alfabético y deben ser ignorados.

A$ se emplea para almacenar las palabras originales, mientras que B$ contiene las letras mayúsculas y minúsculas convertidas, excluyendo los caracteres que no son letras. B$ se ordena mediante una clasificación por burbujas (línea 1100) y A$ sigue por simpatía. Se comprueba que cada palabra tiene las características correctas antes de aceptarla para su clasificación.

```
10 REM Clasificacion de palabras
20 DEFINT N,P,Q,T
30 INPUT "Numero de palabras";N: IF N<=0 THEN 30
40 GOSUB 1000: GOSUB 2000: END
1000 N=N-1: DIM A$[N],B$[N]
1010 FOR Q=0 TO N: PRINT "Palabra";Q+1;" es ";: INPUT Z$: L=LEN(Z$)
1020 A$[Q]=Z$: B$[Q]=Z$
1030 FOR P=1 TO L: T=1: X$=MID$(Z$,P,1)
1040 IF X$="-" OR X$="'" OR X$>="A" AND X$<="Z" OR X$>="a" AND
X$<="z" THEN NEXT ELSE PRINT Z$;" no es una palabra": Q=Q-1: GOTO 1090
1050 FOR P=1 TO L: X$=MID$(Z$,P,1)
1060 IF X$>="A" AND X$<="Z" THEN MID$(B$[Q],T,1)=LOWER$(X$): T=T+1
ELSE IF X$>="a" AND X$<="z" THEN MID$(B$[Q],T,1)=X$: T=T+1
1070 NEXT
1080 B$[Q]=LEFT$(B$[Q],T-1)
1090 NEXT
1100 Q=0: FOR P=0 TO N-1
1110 IF B$[P+1]<B$[P] THEN SW$=B$[P]: B$[P]=B$[P+1]: B$[P+1]=SW$:
SW$=A$[P]: A$[P]=A$[P+1]: A$[P+1]=SW$: Q=Q+1
1120 NEXT: IF Q<>0 THEN 1100 ELSE RETURN
2000 FOR P=0 TO N: PRINT A$[P]: NEXT
2010 RETURN
```

```
run                             (resultado)
Numero de palabras? 9
Palabra  1  es  ? El            El
Palabra  2  es  ? veloz         el
Palabra  3  es  ? zorro         marron
Palabra  4  es  ? salta         perezoso
Palabra  5  es  ? sobre         perro
Palabra  6  es  ? el            salta
Palabra  7  es  ? perezoso      sobre
Palabra  8  es  ? perro         veloz
Palabra  9  es  ? marron        zorro
```

85. Análisis estadístico

Esta subrutina calcula las propiedades estadísticas de un grupo de lecturas (media, desviación típica, mínimo, máximo y número de lecturas en cada intervalo del histograma) y muestra los resultados en forma de histograma 3-D.

Para que la presentación sea satisfactoria, se elige para el histograma un número de intervalos próximo a la raíz cuadrada del número de lecturas.

Las ocho primeras líneas calculan el mínimo, el máximo, la media y la desviación típica de los datos. Los resultados se almacenan en Z$ para controlar el número de dígitos impresos en la presentación (dividiendo). D[2,I+3] se usa con dos fines; en primer lugar, para almacenar el valor menor de los intervalos del histograma en D[1,I+3], y, en segundo lugar, para almacénar el número de valores de cada intervalo en D[2,I+3]. El número más alto de valores es D2MAX.

A[2,I+4] contiene los valores que van a figurar en el histograma, el cual está hecho a escala para que se ajuste a una línea con una pendiente de 1/3 y que pasa por (112,34) y (352,107). Estos dos puntos son iguales a los valores mínimo y máximo. La altura de la línea se ha establecido en 100 en la línea 2030.

Las siguientes líneas dibujan el histograma y lo sombrean. Después de la línea 2090 se dibujan los ejes principales y se imprime la información estadística en la pantalla.

```
10 REM Analisis estadistico
20 BORDER 5: INK 0,1: INK 1,15
30 DEFINT I,P,Q: DEF FN A$(A)=MID$(STR$(A),2)
40 INPUT "Cuantas lecturas (min 4)";N: IF N<4 THEN 40
50 CLS: DIM V[N]
60 PRINT: PRINT "Escriba los valores de las lecturas"
70 PRINT: FOR P=1 TO N: PRINT TAB(5);"V";FN A$(P);"=";: INPUT V[P]: NEXT
80 REM Alternativamente, use la rutina de introduccion de datos (variable sencilla)
para establecer N y V(N)
90 CLS: GOSUB 1000: GOSUB 2000: END
1000 I=INT(SQR(N)): MEAN=0: S=0: XMAX=V[1]: XMIN=V[1]
```

```
1010 FOR P=1 TO N
1020 MEAN=MEAN+V[P]: IF V[P]>XMAX THEN XMAX=V[P]
1030 NEXT: MEAN=MEAN/N
1040 FOR P=1 TO N: D=V[P]-MEAN
1050 S=S+D*D: IF V[P]<XMIN THEN XMIN=V[P]
1060 NEXT: S=SQR(S/(N-1))
1070 T=(XMAX-XMIN)/I: XR=T*I
1080 DIM Z$[I+9],D[2,I+3]
1090 FOR P=6 TO I+7: Z$(P)=STR$(XMIN+(P-6.5)*T): NEXT
1100 Z$(1)=STR$(MEAN)
1110 Z$(2)=STR$(XMIN)
1120 Z$(3)=STR$(XMAX)
1130 Z$(4)=STR$(S)
1140 Z$(5)=""
1150 D2MAX=0: D[2,1]=0
1160 FOR P=1 TO N: FOR Q=6 TO I+7
1170 IF V[P]<=VAL(Z$(Q)) THEN D[2,Q-5]=D[2,Q-5]+1: GOTO 1190
1180 D[2,I+3]=0: NEXT Q
1190 NEXT P
1200 FOR P=1 TO I+2
1210 D[1,P]=VAL(Z$(P+5)): IF D[2,P]>=D2MAX THEN D2MAX=D[2,P]
1220 NEXT P
1230 DIM A[2,I+4]: FOR P=2 TO I+4
1240 A[1,P]=((D[1,P-1]-XMIN)*251/XR+117)*COS(0.32175)
1250 A[2,P]=A[1,P]*TAN(0.32175): NEXT
1260 RETURN
2000 FOR P=2 TO I+3: Q=0
2010 IF A[1,P]+Q>A[1,P+1] THEN 2090
2020 IF P<=I+2 THEN PLOT 112+A[1,P]+Q,34+A[2,P]+Q*TAN(0.32175)
2030 DRAWR O,100/D2MAX*D[2,P]
2040 IF D[2,P]>=D[2,P-1] THEN DRAWR -50,0: GOTO 2070
2050 IF P<I+3 AND Q<50 AND Q/3<(D[2,P-1]-D[2,P])*100/D2MAX THEN
DRAWR -Q,0: GOTO 2060 ELSE DRAWR -50,0
2060 IF D[2,P]<=D[2,P-1] THEN GOTO 2080
2070 IF Q=0 THEN DRAWR 0,-100/D2MAX*(D[2,P]-D[2,P-1])
2080 Q=Q+3: GOTO 2010
2090 NEXT
2100 PLOT 112,34: DRAWR 529,529*TAN(0.32175)
2110 PLOT 112,34: DRAWR -110,0
2120 PLOT 112+A[1,2],34+A[2,2]: DRAWR -50,0
2130 FOR P=1 TO I+1: LOCATE 1,3+P: PRINT LEFT$(Z$(P+5),5);"-";
LEFT$(Z$(P+6),5);" "D[2,P+1]: NEXT
2140 LOCATE 25,4: PRINT "Desv. tipica=";LEFT$(z$(4),7)
2150 LOCATE 26,5: PRINT "Media=";LEFT$(Z$(1),7)
2160 LOCATE 18,21: PRINT LEFT$(Z$(2),5)
2170 LOCATE 33,16: PRINT LEFT$(Z$(3),5)
```

```
2180 PLOT 224,68: DRAWR 50,0
2190 PLOT 464,148: DRAWR 50,0
2200 PLOT 112+((MEAN-XMIN)*255/XR+112)*COS(0.32175),34+
((MEAN-XMIN)*255/XR+112)*SIN(0.32175): DRAWR 0,200
2210 LOCATE 11,1: PRINT "ANALISIS ESTADISTICO"
2220 IF INKEY$="" THEN 2220
2230 RETURN
```

Ejemplo

La figura muestra la presentación de los datos siguientes:

V1 =25.6	V15=35.0	V29=27.9	V43=28.9
V2 =28.3	V16=27.1	V30=32.1	V44=32.0
V3 =30.1	V17=29.3	V31=28.7	V45=27.9
V4 =26.9	V18=30.6	V32=29.1	V46=31.0
V5 =37.6	V19=30.5	V33=32.1	V47=30.5
V6 =30.8	V20=29.5	V34=31.3	V48=29.9
V7 =26.0	V21=28.0	V35=30.0	V49=30.1
V8 =29.3	V22=33.1	V36=29.6	V50=30.3
V9 =30.2	V23=36.0	V37=28.7	V51=28.7
V10=31.6	V24=29.8	V38=35.0	V52=31.2
V11=28.7	V25=31.2	V39=31.2	V53=33.1
V12=29.5	V26=30.1	V40=27.9	V54=28.9
V13=30.3	V27=28.7	V41=30.5	V55=27.9
V14=32.4	V28=31.5	V42=31.6	V56=31.0

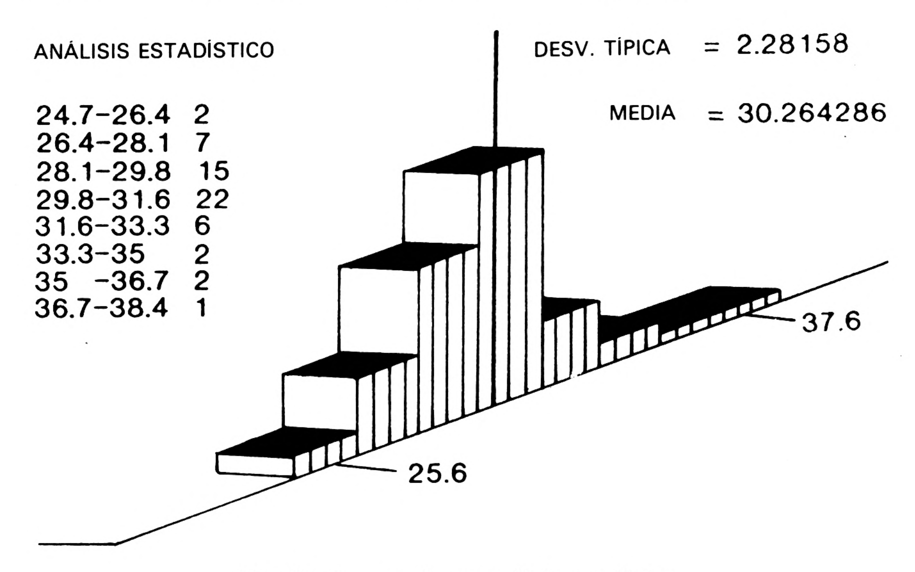


Fig. 85. Imagen de un análisis estadístico.

86. Almacenamiento de cadenas

El almacenamiento de datos, particularmente cuando se hace en forma de números de punto flotante, puede emplear gran cantidad de memoria, y a veces es preferible almacenarlos en forma de cadenas o de matrices literales.

Hay tres casos que deben ser tomados en consideración. En primer lugar, los casos en los que todas las cadenas tienen la misma longitud, como por ejemplo los bytes de código de máquina en hexadecimal. Aquí no son necesarios los separadores. Sin embargo, si los datos tienen longitudes diferentes sí hacen falta los separadores (comas, etc.). En las situaciones en las que la mayoría de las cadenas tienen la misma longitud pero hay algunas más pequeñas, compensa más rellenar éstas con espacios y omitir los separadores.

El primer caso es trivial, pero los dos programas siguientes podrían ser de utilidad cuando se trata de ahorrar memoria.

```
10 REM La misma longitud para todas las cadenas
20 INPUT "Longitud de las cadenas";N: B$=""
30 INPUT "Cadena";A$: IF LEN(A$)>N THEN 30: REM Para terminar introduzca
"="
40 IF A$<>"=" THEN B$=B$+STRING$(N-LEN(A$),32)+A$: GOTO 30
50 GOSUB 2000: END
2000 PRINT B$
2010 RETURN
2020 REM Cambio de la linea 40 por
2030 REM "IF A$<>"=" THEN B$=B$+STRING$(N-LEN(A$),32)+A$+NL$:
GOTO 30" (donde NL$=CHR$(10)+CHR$(13))
2040 REM Impresion en una columna vertical.
2050 REM O cambio de la linea 2000 por
2060 REM "FOR P=1 TO T: PRINT MID$(B$,1+N*(P-1),N): NEXT" (donde T es
el numero de cadenas) hace lo mismo.
```

```
10 REM Almacenar en forma de cadena con una / entre cada elemento
20 DEFSTR N
30 INPUT N: REM N=n1/n2/n3/n4/n5/n6/ y debe acabar con una barra (/). N.B.
No se permiten las comas y las comas invertidas no son necesarias.
40 GOSUB 2000: END
2000 REM Para recuperar los datos
2010 A=1: B=1
2020 FOR P=A TO LEN(N$)
2030 IF MID$(N$,P,1)="/" THEN A=P: PRINT MID$ (N$,B+(B=2),A-B-(B=2))
2040 B=A+1: NEXT
```

87. Impresión con TAG

En modo 2 (MODE 2), PRINT imprime los caracteres ASCII dentro de un área rectangular de 8×8 pixels de tamaño, por lo que resultan bastante difíciles de leer. Empleando TAG, la impresión se produce en la posición del cursor gráfico y, si usted lo prefiere, es posible dejar algo más de espacio entre las letras.

En modo 1 (MODE 1) se puede emplear la forma inversa, es decir, dejar menos espacio entre las letras.

```
10 REM Impresion con TAG-MODE 1
20 MODE 1
30 INPUT "Posicion de impresion (coordenadas x,y)"; X,Y
40 INPUT "Texto que hay que escribir";A$
50 GOSUB 2000: END
2000 CLS: FOR P=1 TO LEN (A$): Z$=MID$(A$,P,1)
2010 MOVE X,Y: TAG
2020 PRINT Z$;: X=X+14+5*(Z$=" ")
2030 IF X>620 THEN X=0: Y=Y-16
2040 NEXT: TAGOFF
2050 RETURN
```

```
10 REM Impresion con TAG-MODE 2
20 MODE 2
30 INPUT "Posicion de impresion (como coordenadas x,y)"; X,Y
40 INPUT "Texto que hay que escribir";A$
50 GOSUB 2000: END
2000 CLS: FOR P=1 TO LEN (A$): Z$=MID$(A$,P,1)
2010 MOVE X,Y: TAG
2020 PRINT Z$;: X=X+10+4*(Z$=" ")
2030 IF X>620 THEN X=0: Y=Y-16
2040 NEXT: TAGOFF
2050 RETURN
```

88. Comprobación de un número binario

Esta prueba es similar a la comprobación de un número decimal, aunque más restringida. La subrutina comprueba un entero binario positivo o negativo y se puede usar, por ejemplo, antes de una rutina de conversión.

```
10 REM Comprobacion de un numero binario
20 PRINT "Escriba el numero en forma de cadena:": PRINT
30 INPUT B$
40 GOSUB 1000: GOSUB 2000: END
1000 FAIL=0: T=1: IF LEFT$(B$,1)="+" OR LEFT$(B$,1)="-" THEN T=2
```

```
1010 IF T=2 AND LEN(B$)=1 THEN FAIL=-1: RETURN
1020 FOR P=T TO LEN(B$)
1030 IF MID$(B$,P,1)<>"0" AND MID$(B$,P,1)<>"1" THEN FAIL=-1:
RETURN ELSE NEXT
1040 RETURN
2000 IF NOT FAIL THEN PRINT "OK": RETURN
2010 PRINT "No es un numero binario.": RETURN
```

89. Comprobación de un número decimal

Como la entrada se realiza en forma de cadena, se puede comprobar cada carácter por separado. Los signos "+" o "−", si aparecen, deben ir al principio, y esto hace que T cambie de 1 a 2, de forma que el signo no está implicado en el bucle de comprobación. *S* cuenta el número de puntos decimales y, si encuentra más de uno de éstos o si los caracteres no se encuentran en el intervalo de 0 a 9, rechaza el número. Las líneas 1010 y 1020 rechazan las entradas que no tienen dígitos.

La subrutina sólo funciona con números decimales positivos o negativos.

```
10 REM Comprobacion de un numero decimal
30 INPUT "Escriba el numero";N$
40 GOSUB 1000: GOSUB 2000: END
1000 FAIL=0: S=0: T=1: IF LEFT$(N$,1)="+" OR LEFT$(N$,1)="-" THEN T=2
1010 SP=INSTR(N$,"."): IF SP=LEN(N$) THEN FAIL=-1: RETURN
1020 IF T=2 AND LEN(N$)=1 THEN FAIL=-1: RETURN
1030 FOR P=T TO LEN(N$): Z$=MID$(N$,P,1)
1040 IF Z$="." THEN S=S+1
1050 IF Z$="." OR Z$>="0" AND Z$<="9" THEN NEXT: IF S<2 THEN RETURN
1060 FAIL=-1
1070 RETURN
2000 IF NOT FAIL THEN PRINT N$: RETURN
2010 PRINT "No es un numero decimal": RETURN
```

90. Cronómetro

Medir la eficacia de un programa en términos de su tiempo de ejecución es muy fácil con la variable de máquina TIME.

La primera instrucción debería colocarse a continuación de cualquier sentencia INPUT, y la segunda después de que el programa haya terminado la impresión o los cálculos.

Para ilustrar su utilización, a continuación se mide el tiempo real empleado por un bucle vacío.

```
10 REM Cronometro
20 INPUT N
```

```
30 T0=TIME
40 FOR P=1 TO N: NEXT: T=(TIME-T0)/300
50 PRINT "El tiempo para";
60 PRINT N;
70 PRINT "bucles es";
80 PRINT ROUND(T,2);"segundos"
```

Ejemplo

```
run
? 10000
El tiempo para 10000 bucles es 10.7 segundos
```

91. Subrayado

Para usar el símbolo de subrayado del teclado es necesario ejecutar una operación OR a través de la opción de impresión transparente (PRINT CHR$(22)+CHR$(1) seguido de CHR$(22)+CHR$(0) para desactivarla). Sin embargo, el resultado no es satisfactorio, ya que el subrayado es la fila inferior de pixels y se mezcla con las letras.

Hay dos alternativas que producen mejores resultados. Impresión con TAG colocará la palabra y el subrayado adecuadamente. No obstante, la siguiente subrutina, que usa un CHR$(208) modificado, también parece ser adecuada.

CHR$(208) tiene asignadas las dos líneas de pixels superiores; la línea 1000 elimina una de ellas. La línea 1010 añade un segundo subrayado si es necesario. SYMBOL AFTER 208 coloca la información del pixel para CHR$(208) en adelante inmediatamente después de HIMEM.

```
10 REM Subrayado
20 SYMBOL AFTER 208: H=HIMEM+2
30 INPUT "Texto que hay que subrayar";A$
40 PRINT "Posicion de impresion": PRINT: INPUT "Fila No.";F: PRINT: INPUT
"Columna No.";C
50 GOSUB 1000: GOSUB 2000: END
1000 POKE H,0: RETURN
1010 REM Para subrayado doble, tambien POKE H+1,255
2000 CLS: LOCATE C,F: PRINT A$
2010 LOCATE C,F+1: FOR P=1 TO LEN(A$): PRINT CHR$(208);: NEXT
2020 REM Para imprimir en la linea que esta justamente debajo de las palabras su-
brayadas, use PRINT CHR$(22);CHR$(1) para activar la opcion de transparencia, y
CHR$(22);CHR$(0) para desactivar la opcion cuando ya no la necesite
2030 REM Alternativamente, proteja el subrayado con una sentencia PRINT al final
de la linea 2010 si no aparece nada impreso inmediatamente debajo de las palabras.
2040 RETURN
```

92. Rotación universal

Esta subrutina permite la rotación, con un ángulo theta, de un conjunto de puntos (que pueden representar un plano o una figura sólida) sobre una línea cualquiera que pasa por dos puntos, (AX,AY,AZ) y (BX,BY,BZ), en el espacio. En la figura, una pirámide doble rota 30° sobre una línea que pasa por el centro de la figura y por un punto (1,1,1), es decir, una línea que surge del centro de una cara triangular.

En la parte de entrada del programa, se introducen las coordenadas (AX,AY,AZ) y (BX,BY,BZ) y después las coordenadas 3-D de las esquinas de la figura, pero cambian antes de ser almacenadas en C[N,2], de forma que el origen se desplaza a (AX,AY,AZ). El ángulo theta se introduce en grados.

En el programa principal, se calcula los desplazamientos DX, DY, DZ de x, y y z entre los dos puntos de la línea, y a continuación el programa pasa a SUB 1110, que establece una matriz unitaria de 3×3. Esto se usa en SUB 1110 para formar la matriz principal de transformación mediante SUB 1200 y SUB 1300. SUB 1200 averigua el ángulo de las diversas rotaciones, y SUB 1300 encuentra en primer lugar la matriz de rotación correcta, A[3,3], que es, para los tres ejes, la siguiente:

$$\underset{LI=1}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & cosA & -sinA \\ 0 & sinA & cosA \end{bmatrix}} \quad \underset{LI=2}{\begin{bmatrix} cosA & 0 & sinA \\ 0 & 1 & 0 \\ -sinA & 0 & cosA \end{bmatrix}} \quad \underset{LI=3}{\begin{bmatrix} cosA & -sinA & 0 \\ sinA & cosA & 0 \\ 0 & 0 & 1 \end{bmatrix}}$$

La última parte de la rutina multiplica A por U y luego transfiere el resultado a U, dejando A y Z disponibles para la siguiente multiplicación.

Por lo que respecta a la secuencia de diagramas para la transformación, éstos ejecutan las operaciones siguientes:

1. Desplazan el origen a (AX,AY,AZ). Esto se lleva a cabo en la entrada (INPUT).
2. Rotan sobre el eje z para colocar a B en el plano z. LI=3; ángulo=ATN(DY/DX); A=−ángulo; B almacena a A.
3. Rotan sobre el eje y para hacer que AB coincida con el eje Z. LI=2; ángulo=ATN(SQR(DX^2+DY^2)/DZ); A=−ángulo; C almacena a A.
4. Rotan theta grados sobre el eje z. LI=3; A=THETA.
5. Rotan en sentido contrario sobre el eje y en un ángulo C. LI=2; A=C.
6. Rotan en sentido contrario sobre el eje z en un ángulo B. LI=3; A=B.

Ahora tenemos la matriz principal de transformación, que es el producto de seis matrices, es decir, $A1^{-1}*A2^{-1}*A3*A2*A1*U$, donde A1, A2 y A3 son las diferentes matrices de rotación y $A1^{-1}$ y $A2^{-1}$ son las inversas de A1 y A2 y giran en dirección contraria. Observe que no es necesario llevar a cabo el proceso formal de inversión, ya que cambiando de signo al ángulo se invierte la trayectoria de rotación. Sin embargo, es fácil mostrar que

$$\begin{bmatrix} cos(-A) & 0 & sen(-A) \\ 0 & 1 & 0 \\ -sen(-A) & 0 & cos(-A) \end{bmatrix} * \begin{bmatrix} cos(A) & 0 & sen(A) \\ 0 & 1 & 0 \\ -sen(A) & 0 & cos(A) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ya que $cos^2A+sen^2A=1$.

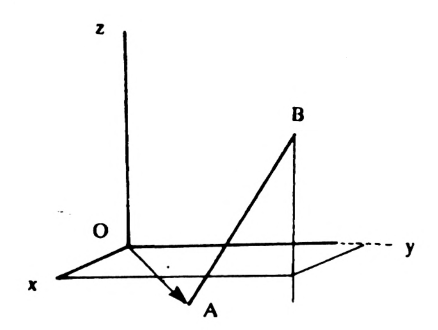
De vuelta de SUB 1110, T[N,2] pasa a estar disponible para almacenar las coordenadas transformadas, obtenidas al multiplicar C por U y al sumar AX, AY, AZ a x, y y z para desplazar el origen a su punto de partida.

En la salida se puede incorporar un factor de amplificación, y la información de PLOT y DRAW se introduce en V$[N,N] para posibilitar que la figura sea dibujada después de usar la fórmula de proyección y de aprobar la relación de aspecto de la pantalla. Para condensar la información sólo se usa el triángulo superior derecho de V$[N,N], pero observe que la línea 2150, NEXT Q,P (NEXT Q: NEXT P), no funciona.

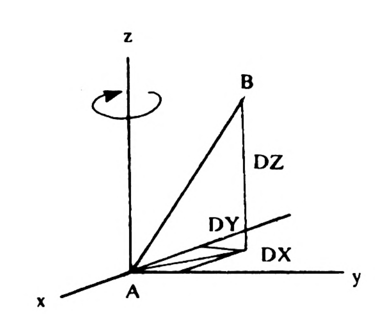
```
10 REM Rotacion universal
20 DEG: DEF FN A$(A)=MID$(STR$(A),2)
30 CLS: PRINT "    Escriba las coordenadas de los dos puntos que definen los ejes
de  rotacion"
40 INPUT "AX";AX: LOCATE 13,VPOS(#0)-1: INPUT "AY";AY: LOCATE
26,VPOS(#0)-1: INPUT "AZ";AZ
50 INPUT "BX";BX: LOCATE 13,VPOS(#0)-1: INPUT "BY";BY: LOCATE
26,VPOS(#0)-1: INPUT "BZ";BZ
60 INPUT "Numero de puntos";N: IF N<=0 THEN 60
70 DIM C[N,2],T[N,2]: CLS
80 PRINT "   Escriba las coordenadas x, y, z de los puntos que deben rotar"
90 FOR P=1 TO N
100 LOCATE 1,P+3: PRINT "x"; FN A$(P);: INPUT Z: C[P,0]=Z-AX
110 LOCATE 14,P+3: PRINT "y"; FN A$(P);: INPUT Z: C[P,1]=Z-AY
120 LOCATE 27,P+3: PRINT "z"; FN A$(P);: INPUT Z: C[P,2]=Z-AZ
130 NEXT
140 INPUT "      OK?     S/N)  ";K$: K$=UPPER$(K$): IF K$="S" THEN CLS
ELSE ERASE C,T: GOTO 60
150 INPUT " Escriba el angulo de rotacion deseado (en grados)";TH
160 GOSUB 1000: GOSUB 2000: END
1000 DX=BX-AX: DY=BY-AY: DZ=BZ-AZ
1010 GOSUB 1100: GOSUB 1110
1020 FOR P=1 TO N
1030 T[P,0]=C[P,0]*U[0,0]+C[P,1]*U[0,1]+C[P,2]*U[0,2]+AX
1040 T[P,1]=C[P,0]*U[1,0]+C[P,1]*U[1,1]+C[P,2]*U[1,2]+AY
1050 T[P,2]=C[P,0]*U[2,0]+C[P,1]*U[2,1]+C[P,2]*U[2,2]+AZ
1060 NEXT : ERASE U
1070 RETURN
1099 REM Matriz unitaria
```

```
1100 DIM U[2,2]: FOR I=0 TO 2: U[I,I]=1: NEXT :RETURN
1109 REM Rotacion universal
1110 GOSUB 1200
1120 LI=2: B=A: A=-A: GOSUB 1300
1130 DY=SQR(DX*DX+DY*DY): DX=DZ: GOSUB 1200
1140 LI=1: C=A: A=-A: GOSUB 1300
1150 LI=2: A=TH: GOSUB 1300
1160 LI=1: A=C: GOSUB 1300
1170 LI=2: A=B: GOSUB 1300
1180 RETURN
1199 REM Angulo
1200 IF ABS(DX)>0.000001 THEN 1240 ELSE A=90
1210 IF DY<0 THEN A=A+90
1220 IF ABS(DY)<0.000001 THEN A=0
1230 RETURN
1240 A=ATN(DY/DX): IF DX<0 THEN A=A+180
1250 RETURN
1299 REM Invertir y multiplicar
1300 DIM A[2,2]: A[LI,LI]=1
1310 A1=(LI+1)MOD 3: A2=(LI+2)MOD 3
1320 CA=COS(A): SA=SIN(A)
1330 A[A1,A1]=CA: A[A2,A2]=CA: A[A1,A2]=-SA: A[A2,A1]=SA
1340 FOR I=0 TO 2: DIM Z[2]: FOR J=0 TO 2: FOR K=0 TO 2
1350 Z[J]=Z[J]+A[J,K]*U[K,I]: NEXT K,J
1360 FOR K=0 TO 2: U[K,I]=Z[K]: NEXT
1370 ERASE Z: NEXT: ERASE A: RETURN
2000 FOR P=1 TO N
2010 PRINT T[P,0];T[P,1];T[P,2]
2020 NEXT
2099 REM Informacion necesaria para una salida grafica, como en el ejemplo
2100 INPUT "Ampliacion";H
2110 DIM V$[N,N]
2120 PRINT "Escriba un "+CHR$(34)+"1"+CHR$(34)+" para representar los
vertices que hay que unir o un "+CHR$(34)+"0"+CHR$(34)+" en caso contrario"
2130 FOR P=1 TO N: FOR Q=P+1 TO N
2140 IF P<>Q THEN PRINT "Vertice";P;"unido al vertice";Q;: INPUT V$(P,Q)
2150 NEXT Q: NEXT P
2160 GOSUB 2500
2170 RETURN
2499 REM Salida grafica
2500 CLS: DIM A[N,1]
2510 FOR P=1 TO N
2520 A[P,0]=320-(58.8*T[P,0]-176.4*T[P,1])*H
2530 A[P,1]=196-(19*T[P,0]+6.5*T[P,1]-162*T[P,2])*H
2540 NEXT
2550 FOR P=1 TO N: FOR Q=P TO N
```

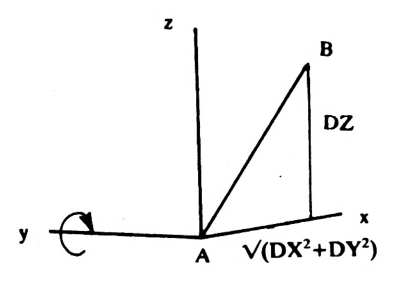
```
2560 IF V$[P,Q]="1" THEN PLOT A[P,0],A[P,1]: DRAW A[Q,0],A[Q,1]
2570 NEXT Q,P
2580 ERASE A,V$
2590 RETURN
```



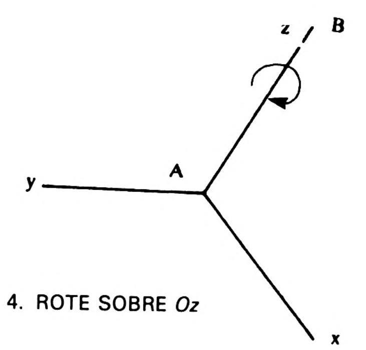
1. DESPLACE EL ORIGEN A *A*



2. ROTE SOBRE *Oz* PARA COLOCAR AB EN EL PLANO *x-z*



3. ROTE SOBRE *Oy* PARA QUE *Oz* COINCIDA CON AB



4. ROTE SOBRE *Oz*

Fig. 92.1. Cómo hacer coincidir el eje Z con la línea

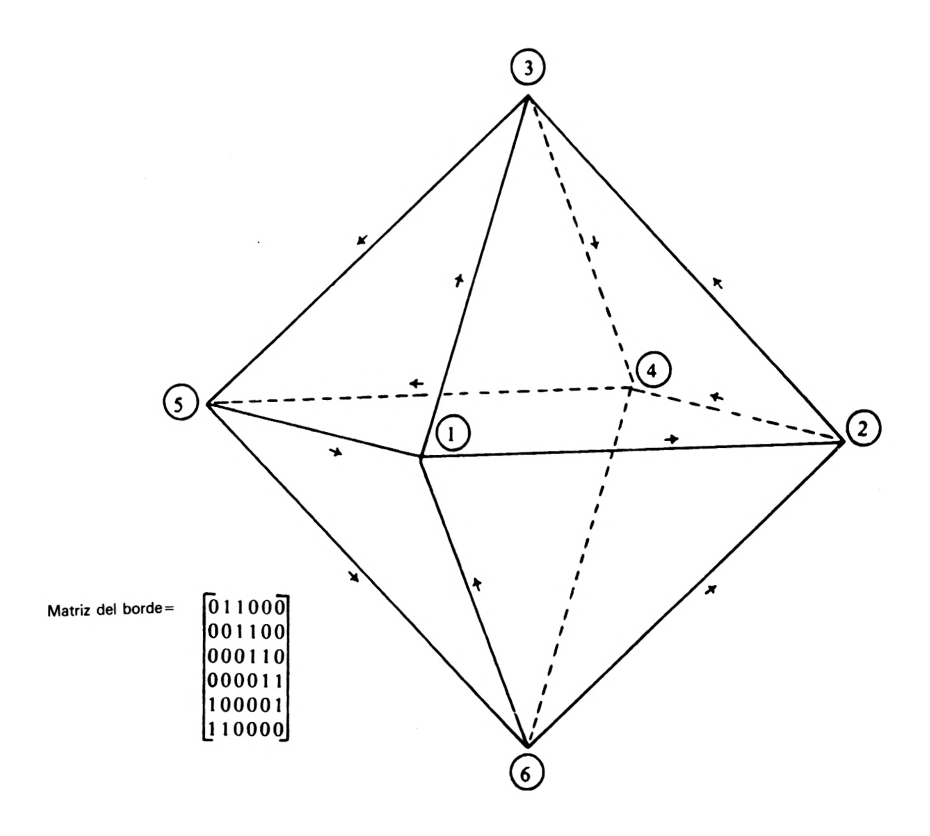


Fig. 92.2. Uso de las matrices para contener la información del borde o de la trayectoria

93. Funciones útiles

En un programa, si se va a emplear una función más de una vez el uso de DEF FN ahorra espacio, dependiendo el ahorro real de la complejidad de la función.

Las funciones que aparecen a continuación son generalmente útiles para cierto número de programas y puede encontrarlas en diversas partes del libro.

```
10 REM PEEK (en una direccion de dos bytes)
20 DEF FN P(P)=PEEK(P)+256*PEEK(P+1)
30 REM POKE (un numero en una posicion de dos bytes)
40 DEF FN Q(Q)=Q-256*INT(Q/256): REM byte bajo
50 DEF FN R(R)=INT(Q/256): REM byte alto
60 REM El mayor de dos numeros
70 DEF FN L(A,B)=A-(B-A)*(B>A)
80 REM o
90 DEF FN L(A,B)=(A+B+ABS(A-B))/2
100 REM Borrar y retroceder N caracteres
110 DEF FN BS$(N)=SPACE$(N)+STRING$(N,CHR$(8))
120 REM Para imprimir o no imprimir segun el resultado de una operacion logica
130 DEF FN FB A$(A$,BOOLE)=MID$(A$,1,-LEN(A$)*BOOLE)
140 REM Impresion de "+" o "-" en una expresion literal
150 DEF FN SIGN$(X)=CHR$(45+2*(X>=0))
160 REM Formato empaquetado. PRINT omite el espacio del signo delante de los
numeros
170 DEF FN A$(A)=MID$(STR$(A),2)
180 REM Formato empaquetado. PRINT omite el espacio delante de los numeros
positivos solo
190 DEF FN A$(A)=MID$(STR$(A),1-LEFT$(STR$(A),1)<>"-",6): REM 6
digitos
200 REM Numeros complejos. Permite imprimir correctamente los numeros
complejos
210 DEF FN A$(A$,A,BOOLE)=MID$(A$,1+A,-LEN(A$)*BOOLE)
220 REM Parentesis para la impresion en algunas circunstancias, por ejemplo para
encerrar numeros negativos
230 DEF FN L$(BOOLE)=CHR$(32-8*BOOLE): DEF FN R$(BOOLE)=CHR$(32-9
*BOOLE)
```

Tabla 93.1. Funciones trigonométricas usadas en sentencias DEF FN

Nombre completo	*Función*	*Función derivada*
Secante	SEC	1/COS(X)
Cosecante	CSC	1/SIN(X)
Cotangente	COT	1/TAN(X)
Arco seno	ARCSIN*	ATN(X/SQR(1-X*X))
Arco coseno	ARCCOS*	-ATN(X/SQR(1-X*X))+1.5708
Arco secante	ARCSEC	ATN(X/SQR(X*X-1))+SGN(SGN(X)-1)*1.5708
Arco cosecante	ARCCSC	ATN(X/SQR(X*X-1))+(SGN(X)-1)*1.5708
Arco cotangente	ARCCOT	ATN(X)+1.5708
Seno hiperbólico	SINH*	(EXP(X)-EXP(-X))/2
Coseno hiperbólico	COSH*	(EXP(X)+EXP(-X))/2
Tangente hiperbólica	TANH*	(EXP(X)-EXP(-X))/(EXP(X)+EXP(-X))
Secante hiperbólica	SECH	2/(EXP(X)+EXP(-X))
Cosecante hiperbólica	CSCH	2/(EXP(X)-EXP(-X))
Cotangente hiperbólica	COTH	(EXP(X)+EXP(-X))/(EXP(X)-EXP(-X))
Arco seno hiperbólico	ARCSINH*	LOG(X+SQR(X*X+1))
Arco coseno hiperbólico	ARCCOSH*	LOG(X+SQR(X*X-1))
Arco tangente hiperbólica	ARCTANH*	LOG((1+X)/(1-X))/2
Arco secante hiperbólica	ARCSECH	LOG((SQR(1-X*X)+1)/X)
Arco cosecante hiperbólica	ARCCSCH	LOG((SGN(X)*SQR(1+X*X)+1)/X)
Arco cotangente hiperbólica	ARCCOTH	LOG((X+1)/(X-1))/2

* *Contiene un nombre reservado (válido para Amstrad, pero no para otros ordenadores)*

N.B. 1.5708=PI/2

Las rutinas

Las siguientes rutinas son programas completos y están listos para que usted los escriba directamente en su Amstrad. Aunque resultan interesantes por sí mismos, también demuestran los usos de varias de las subrutinas que aparecen en este libro.

1. Anagrama

Este programa obtiene todos los anagramas de una palabra, teniendo en cuenta en la solución correcta todas las letras conocidas.

El programa busca primero todas las letras conocidas y las elimina de C$ (líneas 140-200). Si las letras de la palabra original no coinciden con las letras conocidas, se supera el bucle en la línea 190 y el programa vuelve a comenzar.

Las líneas 220-360 efectúan permutaciones de C$. Dichas permutaciones se imprimen en la línea 260 o, si algunas letras son conocidas y una vez que se han vuelto a introducir, en la línea 190, ya que sólo sufren permutaciones las letras desconocidas.

```
10 REM Anagrama
20 DEFINT C,L,P,X,Z: NL$=CHR$(10)+CHR$(13)
30 CLS: Z=0: LINE INPUT "Escriba la palabra  ";A$: L=LEN(A$): GOSUB 400:
GOSUB 370: IF Z=1 THEN 30
40 B$="": C$=A$
50 PRINT NL$;"Conoce las posiciones de todas las letras?";NL$: INPUT
"(s/n)";K$
60 K$=UPPER$(K$): IF K$="N" THEN 220
70 CLS: LOCATE 2,5: PRINT A$: LOCATE 1,8: PRINT "  Escriba una por una las
letras conocidas de la palabra en sus posiciones correctas o escriba una
interrogacion si no conoce la posicion de la letra.";NL$;NL$;CHR$(32);STRING$
L,"*")
80 FOR P=1 TO L
90 LOCATE P+1,20: LINE INPUT X$: IF X$="" THEN 90
100 B$=B$+LEFT$(X$,1): LOCATE P+1,12
110 IF LEFT$(X$,1)="?" THEN PRINT "?";: GOTO 130
120 PRINT LEFT$(X$,1)
130 NEXT P: GOSUB 370: D$=B$: CLS: IF Z=1 THEN 30
140 FOR P=1 TO L
150 IF MID$(B$,P,1)="?" THEN 200
160 FOR Q=1 TO LEN(C$)
170 IF MID$(B$,P,1)<>MID$(C$,Q,1) THEN 190
```

```
180 C$=LEFT$(C$,Q-1)+MID$(C$,Q+1): GOTO 200
190 NEXT Q: IF Q=LEN(C$)+1 THEN 210
200 NEXT P: IF LEN(C$)=0 THEN PRINT A$: END ELSE GOTO 220
210 PRINT NL$; "Su informacion es inconsistente. Pulse cualquier tecla para volver
a empezar.": GOSUB 350: CLS: GOTO 30
220 C=LEN(C$)-1: DIM A[C],B[C]
230 FOR P=1 TO C: A[P]=1: B[P]=C-P+2: NEXT
240 FOR P=1 TO C: IF A[P]>B[P] THEN A[P]=1
250 NEXT P: T=1
260 IF K$="N" OR K$="n" THEN PRINT C$+SPACE$(7-C);: GOTO 300
270 FOR P=1 TO L
280 IF MID$(B$,P,1)="?" THEN MID$(D$,P,1)=MID$(C$,T,1): T=T+1
290 NEXT P: PRINT D$+SPACE$(8-L);
300 FOR X=C TO 1 STEP -1
310 Z$=LEFT$(C$,1): FOR P=1 TO 1+C-X
320 MID$(C$,P,1)=MID$(C$,P+1,1): NEXT P
330 MID$(C$,P,1)=Z$: A[X]=A[X]+1
340 IF A[X]>B[X] THEN NEXT X: END ELSE 240
350 IF INKEY$="" THEN 350
360 RETURN
370 PRINT NL$;NL$;"    Es correcto? (s/n)";: INPUT K$
380 K$=UPPER$(K$): IF K$="S" THEN RETURN
390 Z=1: RETURN
400 FOR P=1 TO L: Z$=MID$(A$,P,1)
410 IF (Z$<"A" OR Z$>"Z") AND (Z$<"a" OR Z$>"z") THEN Z=1: P=L
420 NEXT P: RETURN
```

Ejemplo

```
run
Escriba la palabra? seater
Es correcto? (s/n) s
Conoces las posiciones de todas las letras? (s/n)? s

seater

Escriba una por una las letras conocidas
de la palabra en sus posiciones correctas
o escriba una interrogacion si no conoce la
posicion de la letra.

******

s????r

Es correcto? (s/n)? s
```

```
seater  saeter  sateer  staeer  steaer
setaer  sateer  staeer  steaer  setaer
seater  saeter  steear  setear  seetar
seetar  setear  steear  seeatr  seeatr
seaetr  saeetr  saeetr
Ready
```

2. Simulación de un bucle de control

En el control automático hay un sensor que mide la cantidad que usted trata de controlar y, si la medida difiere del valor ideal, se realizan algunos cambios para que vuelva a dicho valor. Normalmente, la razón de corrección es proporcional a la señal de error, pero se puede añadir términos integrales y derivados para poner a prueba la acción de control.

Una pérdida de tiempo en un bucle de control puede acabar con su estabilidad y desembocar en condiciones de persecución o huida. Un movimiento de piezas mal articuladas en los mecanismos de transmisión o un intervalo de tiempo transcurrido entre la medida y el punto de control son las causas habituales de este problema.

En el siguiente programa se simula la estabilidad de control de una operación de máquina para encontrar el mayor porcentaje de corrección con el que obtener un bucle estable.

Como no es posible realizar la medida en el punto en que funciona la máquina, la medida del tamaño se toma diametralmente opuesta a la posición de la herramienta, y esto introduce un retraso de medio ciclo en el bucle de control.

Para comprobar la estabilidad, se introduce un cambio de paso en el sistema y se calcula su respuesta. En la práctica, puede que quede sin concluir un fragmento de la herramienta o que el ajuste de la misma sea alterado accidentalmente, pero los cambios lentos pueden considerarse como un conjunto de pequeños cambios de paso.

En la primera parte del ciclo no tiene lugar ningún cambio de medida, ya que el tamaño nuevo todavía no ha llegado al punto de medida. Una vez alcanzado este punto, queda un error constante para un medio ciclo posterior y el sistema de control realiza una corrección lineal. Después de aplicar el tamaño corregido, el control cambia a una función cuadrada, después a una cúbica, etc. El que esto sea estable o no depende de la cantidad de corrección llevada a cabo en el segundo medio ciclo después del cambio de paso, es decir, depende de si la corrección es excesiva o insuficiente.

Para investigar esto, sólo tiene que cambiar K en la línea 60 y observar el gráfico de la posición de la herramienta en comparación con el tiempo. Esto puede variar desde el tipo sobreamortiguado (K=.01), pasando por el amortiguado crítico (K=.018) hasta llegar al frenéticamente oscilante.

La simulación indica que el sistema funciona mejor si sólo se corrige el 37% del error en la mitad de un ciclo (K=.018).

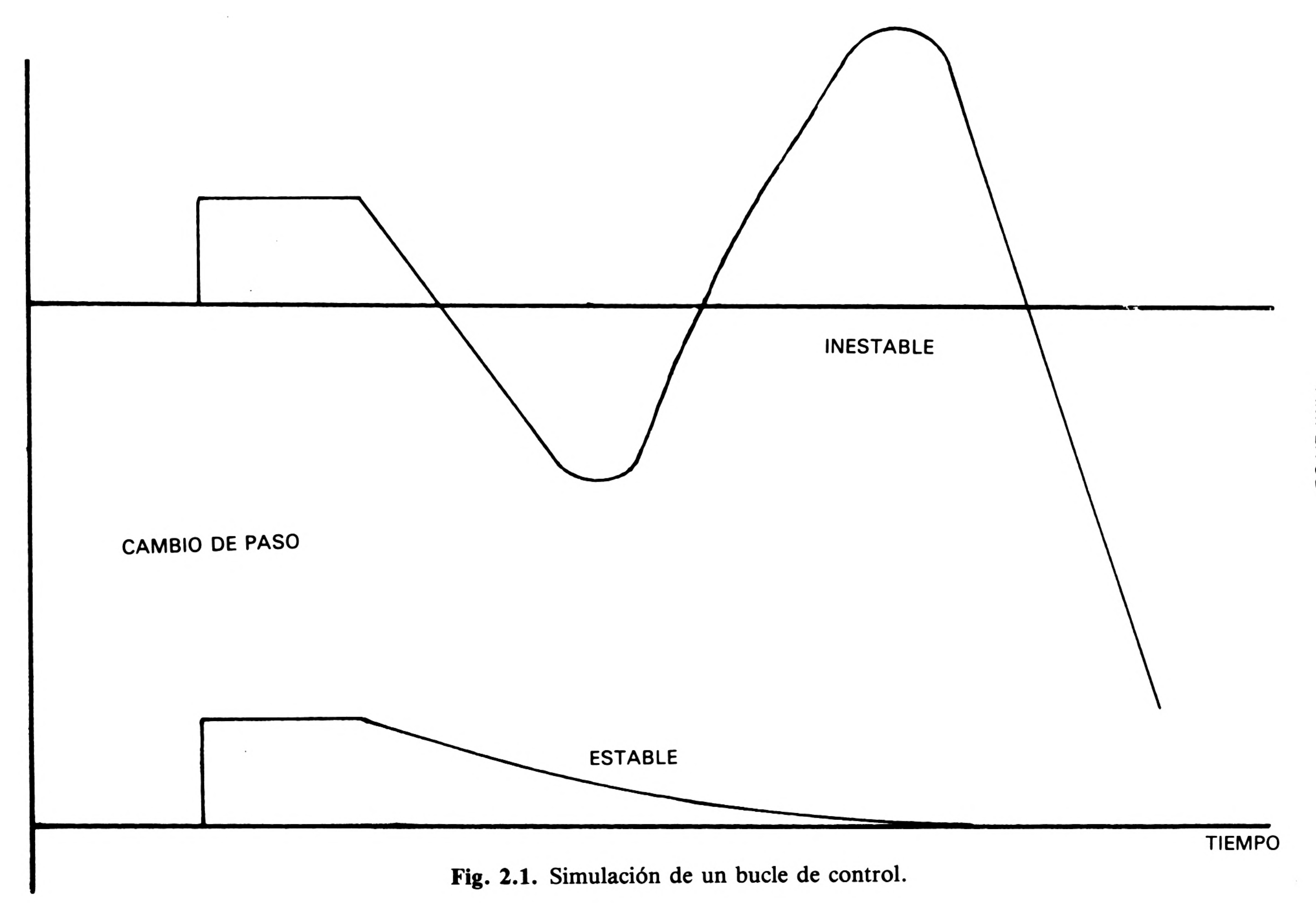


Fig. 2.1. Simulación de un bucle de control.

```
10 REM Estabilidad de un bucle de control
20 MODE 1: BORDER 1: INK 0,15: INK 1,9: INK 2,0: INK 3,6
30 SYMBOL AFTER 208: POKE HIMEM+1,0
40 DEF FN H(T,X,Y)=-(X<=T)+(Y<=T): REM Operador Heaviside
50 CLS: PEN 2: PAPER 3: PRINT SPC(9);"Estabilidad del bucle de control";
SPC(9): PRINT SPC(9);: FOR P=1 TO 22: PRINT CHR$(208);: NEXT: PRINT
SPC(9): PRINT
60 PAPER 0: PRINT " Este programa demuestra el funcionamiento de un bucle de
control que tiene una perdida de tiempo, de forma que el porcentaje de correccion
del error determina la estabilidad del sistema."; SPC(29)
70 PRINT "      Introduciendo valores diferentes de la constante de proporcionali-
dad, K, puede usted ver como afecta esto a la respuesta produciendo un cambio
de paso. K deberia estar en el margen de 0-0.1 para mostrar el efecto completo.";
SPC(16)
80 PAPER 3: PRINT: PRINT " Constante de proporcionalidad": PRINT: INPUT
"K";K
90 RO=100: DR=20: W=0.1: T1=PI/W: Z=K*T1
100 CLS
110 PRINT: PRINT SPC(11);"Condiciones iniciales": PRINT SPC(11);: FOR P=1 TO
18: PRINT CHR$(208);: NEXT
120 PAPER 0: PRINT: PRINT TAB(3);"Radio";TAB(16) USING "###.####";RO
130 PRINT: PRINT TAB(3);"Paso";TAB(16) USING "###.####";DR
140 PRINT: PRINT TAB(3);"Medio ciclo";TAB(16) USING "###.####";T1
150 PRINT: PRINT TAB(3);"Constante K";TAB(16) USING "###.####";K
160 PRINT: PRINT "     Listo?      S/N)";: INPUT K$: K$=UPPER$(K$): IF
K$<>"S" THEN CLS: GOTO 80
170 T2=T1+T1: T3=T2+T1: T4=T3+T1: T5=T4+T1: T6=T5+T1: T7=R6+T1:
REM Periodos de medio ciclo
180 TAG: MOVE 576,47: PRINT "AXIS": CLS : TAGOFF
190 FOR T=1 TO T3
200 R=RO+FN H(T,T1,T2)*DR+FN H(T,T2,T3)*DR*(1-K*(T-T2))
210 GOSUB 340: NEXT T
220 FOR T=T3 TO T4: C=K*(T-T3)
230 R=RO+DR*(1-Z-C+C↑2/2)
240 GOSUB 340: NEXT T
250 FOR T=T4 TO T5: C=K*(T-T4)
260 R=RO+DR*(1-2*Z+Z↑2/2-(1-Z)*C+C↑2/2-C↑3/6)
270 GOSUB 340: NEXT T
280 FOR T=T5 TO T6: C=K*(T-T5)
290 R=RO+DR*(1-3*Z+2*Z↑2-Z↑3/6-C*(1-2*Z+Z↑2/2)+C↑2*(1-Z)
/2-C↑3/6+C↑4/24)
300 GOSUB 340: NEXT T
310 FOR T=T6 TO T7: C=K*(T-T6)
320 R=RO+DR*(1-4*Z+9*Z↑2/2-4+Z↑3/3+Z↑4/24-C*(1-3*Z+2*Z
↑2-Z↑3/6)+C↑2*(1-2*Z+Z↑2/2)/2-C↑3*(1-Z)/6+C↑4/24-C↑5/12)
330 GOSUB 340: NEXT T: GOTO 350
```

```
340 PLOT 3*T,40: DRAWR 0,R: RETURN
350 LOCATE 1,24: PEN 2: END
```

3. Cristales cúbicos

Las caras de un cristal se pueden designar mediante tres números enteros, h, k y l (que son los recíprocos de las intersecciones de la cara en los ejes x, y y z). En el cristal perfectamente simétrico, los índices de Millar, que es el nombre de estos números, son positivos y negativos, y permutados por la simetría para dar un grupo de caras que encierra el perfil del cristal.

Por tanto, (100), es decir, h, k y l, se convierten en $(\bar{1}00)$, (010), $(0\bar{1}0)$, (001) y $(00\bar{1})$ que encierran un cubo. (111) tiene ocho permutaciones y da una pirámide doble; (110) tiene doce y da un dodecaedro rómbico, etc.

El programa encuentra primero una matriz de control, C[26,3], que permitirá dibujar las líneas correctas, y después pide los valores de h, k y l que usted quiere introducir. Calcula las coordenadas de los vértices de la figura y los une en una proyección tridimensional usando líneas continuas para el frente y líneas punteadas para la parte de atrás.

La figura está etiquetada con la forma {*hkl*} y se puede insertar los siguientes valores pulsando inicialmente cualquier tecla seguida por los valores h, k y l nuevos.

Las líneas 100-220 establecen una matriz de $3 \times 8 \times 6$ que contiene todas las permutaciones de h, k y l con los dos valores, positivo y negativo.

Las líneas 240-245 contienen la información necesaria para permitir seleccionar 26 tripletes de 48. Éstos se almacenan en A[26,3,3] a través de las líneas 250-290. Las subrutinas 1200, 1210 y 1220 seleccionan las partes de esta matriz que contienen valores diferentes de H, K y L (líneas 10, 20 y 30) para formar la matriz C[26,3].

Después de los tres ciclos controlados por G, se piden los valores que se desea dar a h, k y l (líneas 40-43) y las líneas 50 y 60 colocan sus valores absolutos en orden descendente.

Para encontrar un vértice, se eligen tres caras que se unen en el vértice y se resuelven sus ecuaciones como un sistema. Esto da las coordenadas del vértice, ya que éste es el único punto común a los tres planos.

Esto se hace en las líneas 300-390, donde se resuelve cada uno de los 26 tripletes para x, y y z, suponiendo que el determinante D no es cero. Los resultados se almacenan en I[26,3].

Las líneas 400-420 usan la fórmula de proyección para encontrar las coordenadas bidimensionales en la pantalla. *H* se usa como un factor de escala.

Las líneas 440-480 separan las caras en siete tipos diferentes:

h	k	l
h	l	l
h	h	l
h	h	h
h	k	0
h	h	0
h	0	0

Las subrutinas 500, 600, 700, 800 y 900 deciden qué puntos hay que unir basándose en la información de la matriz de control. Por ejemplo, en un cubo sólo deben unirse las esquinas adyacentes, y no las líneas diagonales de las caras o del cuerpo. SUB 1500 sirve para las líneas continuas y SUB 1600 para las punteadas.

Para ayudarle cuando esté escribiendo el programa, se ha incluido el organigrama principal y el diagrama de la separación de las caras del cristal, así como la matriz de control C[26,3] y la matriz de tripletes de las caras A[26,3,3].

```
1 REM Cristal cubico de clase 4/m3m–Holosimetrico
2 BORDER 13: INK 0,2: INK 1,0
3 DEFINT B–D,G,H,K,L,N,R–Z: DEF FN A$(A)=MID$(STR$(A),2)
10 DIM C[26,3]: G=0: H=1: K=1: L=1: GOTO 100
20 ERASE A: G=G+1: IF G=1 THEN L=0: GOTO 100
30 IF G=2 THEN K=0: GOTO 100
40 PRINT "Forma siguiente"
41 INPUT "h=";H: H=ABS(H)
42 INPUT "k=";K: K=ABS(K)
43 INPUT "l=";L: L=ABS(L): IF H+K+L=0 THEN 40
50 IF K>H THEN SW=K: K=H: H=SW
60 IF L>K THEN SW=L: L=K: K=SW: GOTO 50
70 Z$="FORMA DE CRISTAL CUBICO "+CHR$(123)+FN A$(H)+FN A$(K)+FN
A$(L)+CHR$(125)
75 CLS
100 DIM A[8,2],D[18],I[3,8,6]
110 X[0]=1: X[1]=1: X[2]=1
120 FOR A=1 TO 8: FOR B=0 TO 2: A[A,B]=X[B]: NEXT
130 IF A=2 OR A=6 THEN X[2]=–X[2]: GOTO 160
140 IF A=4 THEN X[0]=–X[0]: X[1]=–X[1]: X[2]=–X[2]: GOTO 160
150 X[1]=–X[1]
160 NEXT
170 X[0]=H: X[1]=K: X[2]=L: N=1
180 FOR B=0 TO 2: D[N]=X[B]: N=N+1: NEXT: SW=X[0]: X[0]=X[2]: X[2]=SW
190 FOR B=0 TO 2: D[N]=–X[B]: N=N+1: NEXT: SW=X[0]: X[0]=X[1]:
X[1]=SW
200 IF N<18 THEN 180
210 FOR B=1 TO 8: N=1
```

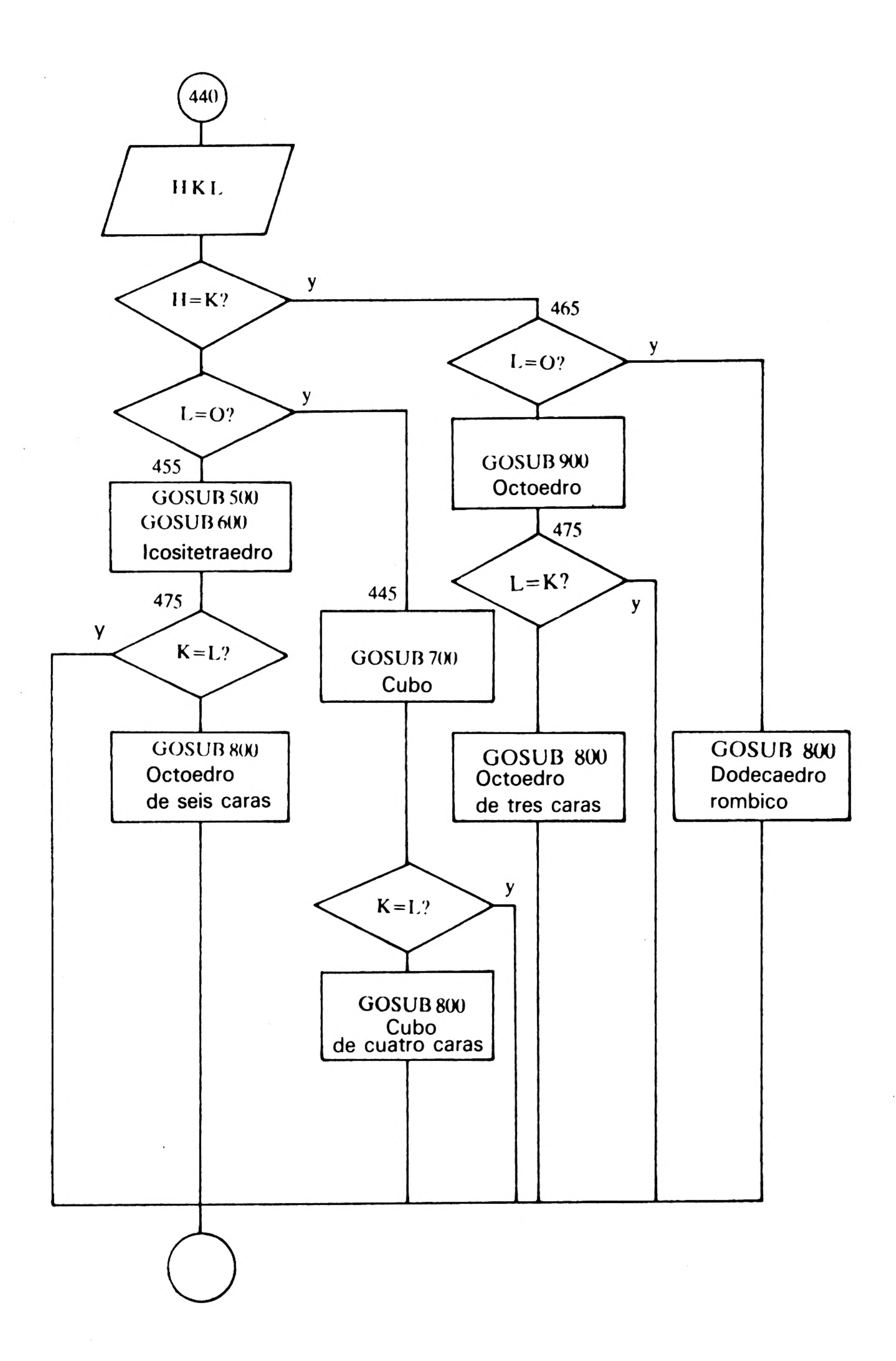
```
220 FOR C=1 TO 6: FOR A=0 TO 2: I[A+1,B,C]=A[B,A]*D[N]: N=N+1: NEXT
A,C,B
230 ERASE A,D: DIM A[26,3,3]
240 A$="118647123456781113335557777"
241 B$="263574123456788886664444222"
242 C$="647118123456784262488626844"
243 D$="136136111111111351351351355"
244 E$="136136333333336426426426422"
245 F$="425425555555551351351351355"
250 FOR C=1 TO 26: FOR A=1 TO 3
260 A[C,1,A]=I[A,VAL(MID$(A$,C,1)),VAL(MID$(D$,C,1))]
270 A[C,2,A]=I[A,VAL(MID$(B$,C,1)),VAL(MID$(E$,C,1))]
280 A[C,3,A]=I[A,VAL(MID$(C$,C,1)),VAL(MID$(F$,C,1))]
290 NEXT A,C: ERASE I
291 IF G=0 THEN GOSUB 1200: GOTO 20
292 IF G=1 THEN GOSUB 1210: GOTO 20
293 IF G=2 THEN GOSUB 1220: GOTO 20
300 DIM I[26,3]: FOR A=1 TO 26
310 R=A[A,1,1]: S=A[A,1,2]: T=A[A,1,3]
320 U=A[A,2,1]: V=A[A,2,2]: W=A[A,2,3]
330 X=A[A,3,1]: Y=A[A,3,2]: Z=A[A,3,3]
340 D=R*(V*Z-Y*W)+S*(W*X-Z*U)+T*(U*Y-X*V)
350 IF D=0 THEN 390
360 I[A,1]=(V*Z-Y*W+S*(W-Z)+T*(Y-V))/D
370 I[A,2]=(R*(Z-W)+W*X-Z*U+T*(U-X))/D
380 I[A,3]=(R*(V-Y)+S*(X-U)+U*Y-X*V)/D
390 NEXT: ERASE A
400 DIM A[26,3]: FOR A=1 TO 26
410 A[A,1]=320-58.8*I[A,1]*H+176.4*I[A,2]*H
420 A[A,2]=196-19*I[A,1]-6.5*I[A,2]*H+162*I[A,3]*H: NEXT: ERASE I
430 MODE 2
440 IF H=K THEN 465 ELSE IF L<>0 THEN 455
445 GOSUB 700: IF K<>L THEN GOSUB 800
450 GOTO 2000
455 GOSUB 500: GOSUB 600: IF K<>L THEN GOSUB 800
460 GOTO 2000
465 IF L<>0 THEN GOSUB 900: GOTO 475
470 GOSUB 800: GOTO 2000
475 IF L<>K THEN GOSUB 800
480 GOTO 2000
500 FOR A=1 TO 6: FOR B=1 TO 3: IF C[A,B]<>0 THEN GOSUB 520
510 NEXT B,A: RETURN
520 FOR C=15 TO 26: IF C[C,B]<>C[A,B] THEN 550
530 IF C[C,1]=-1 OR A=4 THEN GOSUB 1600: GOTO 550
540 GOSUB 1500
550 NEXT C: RETURN
```

```
600 FOR A=7 TO 14: FOR C=15 TO 26
610 IF (C[C,1]=C[A,1] OR C[C,1]=0) AND (C[C,2]=C[A,2] OR C[C,2]=0) AND
(C[C,3]=C[A,3] OR C[C,3]=0) THEN GOSUB 630
620 NEXT C,A: RETURN
630 IF A>10 AND C>19 THEN GOSUB 1600: GOTO 650
640 GOSUB 1500
650 RETURN
700 FOR A=7 TO 14 STEP 2: FOR C=7 TO 14
710 IF C[A,1]=-C[C,1] AND C[A,2]=C[C,2] AND C[A,3]=C[C,3] OR
C[A,1]=C[C,1] AND C[A,2]=-C[C,2] AND C[A,3]=C[C,3] OR C[A,1]=C[C,1]
AND C[A,2]=C[C,2] AND C[A,3]=-C[C,3] THEN GOSUB 730
720 NEXT C,A: RETURN
730 IF C=14 THEN GOSUB 1600: GOTO 750
740 GOSUB 1500
750 RETURN
800 FOR A=7 TO 14: FOR B=1 TO 3: FOR C=1 TO 6: IF C[A,B]=C[C,B] THEN
GOSUB 820
810 NEXT C,B,A: RETURN
820 IF A>10 THEN 840
830 GOSUB 1500: GOTO 860
840 IF C[A,2]=-1 OR C[C,1]=-1 THEN GOSUB 1600: GOTO 860
850 GOTO 830
860 RETURN
900 FOR A=1 TO 6: FOR B=1 TO 3: IF C[A,B]<>0 THEN GOSUB 920
910 NEXT B,A: RETURN
920 FOR C=1 TO 6: IF B=3 THEN 950
930 IF C[C,B+1]<>0 THEN 970
940 GOTO 990
950 IF C[C,1]<>0 THEN 970
960 GOTO 990
970 IF C=4 OR A=4 THEN GOSUB 1600: GOTO 990
980 GOSUB 1500
990 NEXT: RETURN
1200 FOR A=7 TO 14: GOSUB 1300: NEXT: RETURN
1210 FOR A=15 TO 26: GOSUB 1300: NEXT: RETURN
1220 FOR A=1 TO 6: GOSUB 1300: NEXT: RETURN
1300 FOR B=1 TO 3: C[A,B]=A[A,1,B]: NEXT: RETURN
1500 PLOT A[A,1],A[A,2]: DRAW A[C,1],A[C,2]: RETURN
1600 IF A[A,1]=A[C,1] THEN 1670 ELSE IF A[A,2]=A[C,2] THEN 1630
1610 P=(A[C,2]-A[A,2])/(A[C,1]-A[A,1])
1620 IF ABS(P)<1 THEN 1640 ELSE P=1/P: GOTO 1680
1630 P=0
1640 I=10*SGN(A[C,1]-A[A,1])/SQR(1+P*P)
1650 FOR Q=0 TO (A[C,1]-A[A,1])/I
1660 PLOT A[A,1]+Q*I,A[A,2]+Q*I*P: DRAWR 0.4*I,0.4*I*P: NEXT:
RETURN
```

```
1670 P=0
1680 I=10*SGN(A[C,2]-A[A,2])/SQR(1+P*P)
1690 FOR Q=0 TO (A[C,2]-A[A,2])/I
1700  PLOT  A[A,1]+Q*I*P,A[A,2]+Q*I:  DRAWR  0.4*I*P,0.4*I:  NEXT:
RETURN
2000 TAG: MOVE 255,380: PRINT Z$;
2010 IF INKEY$="" THEN 2010
2020 TAGOFF: GOTO 20
```

N	C[26,3]	A[26,3,3]		
1	100	HKL	$H\bar{K}L$	$H\bar{L}\bar{K}$
2	001	KLH	$\bar{K}LH$	$\bar{L}\bar{K}H$
3	010	KHL	$\bar{K}HL$	$\bar{L}H\bar{K}$
4	$\bar{1}00$	$\bar{H}KL$	$\bar{H}\bar{K}L$	$\bar{H}\bar{L}\bar{K}$
5	$00\bar{1}$	$KL\bar{H}$	$\bar{K}L\bar{H}$	$\bar{L}\bar{K}\bar{H}$
6	$0\bar{1}0$	$K\bar{H}L$	$\bar{K}\bar{H}L$	$\bar{L}\bar{H}\bar{K}$
7	111	HKL	KLH	LHK
8	$1\bar{1}1$	$H\bar{K}L$	$K\bar{L}H$	$L\bar{H}K$
9	$1\bar{1}\bar{1}$	$H\bar{K}\bar{L}$	$K\bar{L}\bar{H}$	$L\bar{H}\bar{K}$
10	$11\bar{1}$	H$K\bar{L}$	$KL\bar{H}$	$LH\bar{K}$
11	$\bar{1}1\bar{1}$	$\bar{H}\bar{K}L$	$\bar{K}\bar{L}H$	$\bar{L}\bar{H}K$
12	$\bar{1}11$	$\bar{H}KL$	$\bar{K}LH$	$\bar{L}HK$
13	$\bar{1}1\bar{1}$	$\bar{H}K\bar{L}$	$\bar{K}L\bar{H}$	$\bar{L}H\bar{K}$
14	$\bar{1}\bar{1}\bar{1}$	$\bar{H}\bar{K}\bar{L}$	$\bar{K}\bar{L}\bar{H}$	$\bar{L}\bar{H}\bar{K}$
15	110	HKL	KHL	$HK\bar{L}$
16	101	KLH	HLK	$K\bar{L}H$
17	011	LHK	LKH	$\bar{L}HK$
18	$1\bar{1}0$	$H\bar{K}\bar{L}$	$K\bar{H}\bar{L}$	$H\bar{K}L$
19	$10\bar{1}$	$K\bar{L}\bar{H}$	$H\bar{L}\bar{K}$	$KL\bar{H}$
20	011	$L\bar{H}\bar{K}$	$L\bar{K}\bar{H}$	$\bar{L}\bar{H}\bar{K}$
21	$\bar{1}\bar{1}0$	$\bar{H}\bar{K}L$	$\bar{K}\bar{H}L$	$\bar{H}\bar{K}L$
22	$\bar{1}01$	$\bar{K}\bar{L}H$	$\bar{H}\bar{L}K$	$\bar{K}LH$
23	$0\bar{1}1$	$\bar{L}\bar{H}K$	$\bar{L}\bar{K}H$	$L\bar{H}K$
24	$\bar{1}10$	$\bar{H}K\bar{L}$	$\bar{K}H\bar{L}$	$\bar{H}KL$
25	$\bar{1}0\bar{1}$	$\bar{K}L\bar{H}$	$\bar{H}L\bar{K}$	$\bar{K}\bar{L}\bar{H}$
26	$01\bar{1}$	$\bar{L}H\bar{K}$	$\bar{L}K\bar{H}$	$LH\bar{K}$

Cristal cúbico de clase 4/m3m
Matriz de control y matriz de triplete de cara


440
HKL
H=K?
y
465
L=O?
y
L=O?
y
455
GOSUB 500
GOSUB 600
Icositetraedro
GOSUB 900
Octoedro
475
L=K?
y
475
445
y
K=L?
GOSUB 700
Cubo
GOSUB 800
Octoedro
de seis caras
GOSUB 800
Octoedro
de tres caras
GOSUB 800
Dodecaedro
rombico
y
K=L?
GOSUB 800
Cubo
de cuatro caras

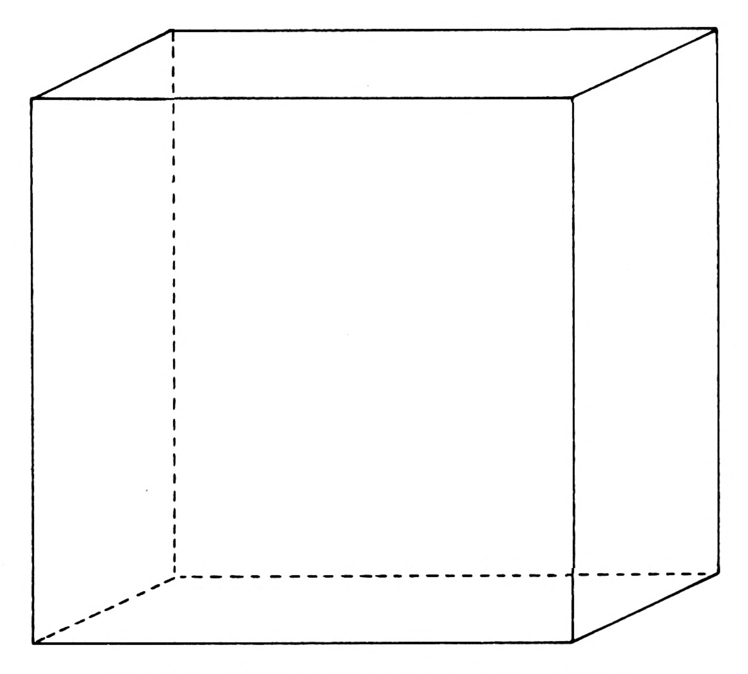


Fig. 3.2. Cristales cúbicos de clases 23 y 4/m3m.
Cubo de forma {100}

4. Forma de cristal 23: ulmanita

El mineral llamado ulmanita pertenece a la clase de cristales cúbicos con la menor simetría. Mientras que la clase con simetría completa tiene 3 ejes cuádruples, 4 ejes triples, 6 ejes dobles y 9 planos de espejo, así como un centro, la clase con la simetría menor sólo tiene 4 ejes triples y 3 ejes dobles. Esto da lugar a varias formas externas diferentes basadas más en el tetraedro que en la pirámide doble, a pesar de que algunas formas, por ejemplo {100}, son cubos en ambas clases.

El programa para la forma 23 (léase 'dos tres') es similar a la forma 4/m3m (léase 'cuatro partido eme tres eme' y muchas de las subrutinas son idénticas. Sin embargo, hay algunos cambios sutiles y debe tener cuidado si deriva un listado del otro.

Los puntos de diferencia principales son:

1. Los valores H, K y L empleados para constituir la matriz de control son 111, −111 y 011, y se usa la tercera columna, no la primera. (Véanse las líneas 20, 30 y 1300.)
2. H, K y L están colocados en orden ascendente (líneas 50 y 60).
3. Las líneas 100-220 constituyen la matriz 3×8×6 de otro modo, pero el resultado es el mismo.

4. Sólo hay 20 vértices posibles en la clase 23, así que 26 se cambia por 20 en las líneas 230, 250, 300 y 400. A$ a F$ contienen 20 caracteres.
5. L se usa como factor de escala en la fórmula de proyección. Obviamente, las subrutinas 500, 600, 700, 800 y 900 son diferentes, como lo son las líneas 400-490, que separan las diferentes caras del cristal.

```
1 REM Cristal cubico de clase 23-Ulmanita
2 BORDER 13: INK 0,2: INK 1,0
3 DEFINT B-D,G,H,K,L,N,R-Z: DEF FN A$(A)=MID$(STR$(A),2)
10 DIM C[20,3],X[2]: G=0: H=1: K=1: L=1: GOTO 100
20 ERASE A: G=G+1: IF G=1 THEN H=-1: GOTO 100
30 IF G=2 THEN H=0: GOTO 100
40 PRINT "Forma siguiente"
41 INPUT "h=";H: H=ABS(H)
42 INPUT "k=";K: K=ABS(K)
43 INPUT "l=";L: L=ABS(L): IF H+K+L=0 THEN 40
50 IF H>K THEN SW=K: K=H: H=SW
60 IF K>L THEN SW=L: L=K: K=SW: GOTO 50
70 Z$="FORMA DE CRISTAL CUBICO "+CHR$(123)+FN A$(H)+FN A$(K)+FN
A$(L)+CHR$(125)
100 DIM A[8,3],D[18],I[3,8,6],P[6,2]
110 X[0]=H: X[1]=K: X[2]=L: N=1
120 FOR A=1 TO 4: A[A,1]=1: A[A+4,1]=-1: NEXT
130 RESTORE 130: DATA 1,-1,-1,1,-1,,1,1,-1
140 FOR A=1 TO 8: READ A[A,2]: NEXT
150 FOR A=1 TO 5 STEP 4: A[A,3]=1: A[A+1,3]=1: A[A+2,3]=-1:
A[A+3,3]=-1: NEXT
160 FOR A=0 TO 6: T=2-(A/2-INT(A/2))*2
170 FOR B=0 TO 2: P[A,B]=X[B]: NEXT
180 SW=X[0]: X[0]=X[T]: X[T]=SW: NEXT
190 FOR A=0 TO 5
200 FOR B=0 TO 2: D[N]=P[A,B]: N=N+1: NEXT: A=A+1
210 FOR B=0 TO 2: D[N]=-P[A,B]: N=N+1: NEXT B,A
220 FOR B=1 TO 8: N=1: FOR C=1 TO 6: FOR A=1 TO 3:
I[A,B,C]=A[B,A]*D[N]: N=N+1: NEXT A,C,B
230 ERASE A,D,P: DIM A[20,3,3]
240 A$="13573175513771353157"
241 B$="13575713157317531375"
242 C$="13577531153717531375"
243 D$="11111111111133335555"
244 E$="33333333111133335555"
245 F$="55555555555511113333"
250 FOR C=1 TO 20: FOR A=1 TO 3
260 A[C,1,A]=I[A,VAL(MID$(A$,C,1)),VAL(MID$(D$,C,1))]
270 A[C,2,A]=I[A,VAL(MID$(B$,C,1)),VAL(MID$(E$,C,1))]
280 A[C,3,A]=I[A,VAL(MID$(C$,C,1)),VAL(MID$(F$,C,1))]
```

```
290 NEXT A,C: ERASE I
291 IF G=0 THEN GOSUB 1200: GOTO 20
292 IF G=1 THEN GOSUB 1210: GOTO 20
293 IF G=2 THEN GOSUB 1220: GOTO 20
300 DIM I[20,3]: FOR A=1 TO 20
310 R=A[A,1,1]: S=A[A,1,2]: T=A[A,1,3]
320 U=A[A,2,1]: V=A[A,2,2]: W=A[A,2,3]
330 X=A[A,3,1]: Y=A[A,3,2]: Z=A[A,3,3]
340 D=R*(V*Z-Y*W)+S*(W*X-Z*U)+T*(U*Y-X*V)
350 IF D=0 THEN 390
360 I[A,1]=(V*Z-Y*W+S*(W-Z)+T*(Y-V))/D
370 I[A,2]=(R*(Z-W)+W*X-Z*U+T*(U-X))/D
380 I[A,3]=(R*(V-Y)+S*(X-U)+U*Y-X*V)/D
390 NEXT: ERASE A
400 DIM A[20,3]: FOR A=1 TO 20
410 A[A,1]=320-58.8*I[A,1]*L++176.4*I[A,2]*L
420 A[A,2]=196-19*I[A,1]-6.5*I[A,2]*L+162*I[A,3]*L: NEXT: ERASE I
430 MODE 2
440 IF K=L THEN 480 ELSE IF H=K THEN 460
450 GOSUB 800: GOSUB 900: GOTO 2000
460 IF H=0 THEN GOSUB 500: GOTO 2000
470 GOSUB 600: GOSUB 700: GOTO 2000
480 IF H=K THEN GOSUB 700: GOTO 2000
490 GOSUB 800: GOTO 2000
500 FOR A=1 TO 4: FOR C=5 TO 8
510 IF C[A,1]=-C[C,1] AND C[A,2]=C[C,2] AND C[A,3]=C[C,3] OR
C[A,1]=C[C,1] AND C[A,2]=-C[C,2] AND C[A,3]-C[C,3] OR C[A,1]=C[C,1]
AND C[A,2]=C[C,2] AND C[A,3]=-C[C,3] THEN GOSUB 530
520 NEXT C,A: RETURN
530 IF C=5 THEN GOSUB 1600: GOTO 550
540 GOSUB 1500
550 RETURN
600 FOR A=1 TO 4: FOR C=5 TO 8
610 IF C<>A+4 AND C[A,1]=-1 THEN GOSUB 1600: GOTO 630
620 IF C<>A+4 THEN GOSUB 1500
630 NEXT C,A: RETURN
700 FOR A=5 TO 7: FOR C=A+1 TO 8
710 IF C[A,1]=-1 AND C[C,1]=-1 THEN GOSUB 1600: GOTO 730
720 GOSUB 1500
730 NEXT C,A: RETURN
800 FOR A=1 TO 8: FOR C=9 TO 20
810 IF (C[C,1]=C[A,1] OR C[C,1]=0) AND (C[C,2]=C[A,2] OR C[C,2]=0) AND
(C[C,3]=C[A,3] OR C[C,3]=0) THEN GOSUB 830
820 NEXT C,A: RETURN
830 IF A=3 OR A=5 OR A=4 AND H<>0 OR C=19 OR C=20 THEN GOSUB
1600: GOTO 850
```

```
840 GOSUB 1500
850 RETURN
900 FOR A=9 TO 20 STEP 2: C=A+1
910 IF C[A,1]=-1 THEN GOSUB 1600: GOTO 930
920 GOSUB 1500
930 NEXT A: RETURN
1200 FOR A=1 TO 4: GOSUB 1300: NEXT: RETURN
1210 FOR A=5 TO 8: GOSUB 1300: NEXT: RETURN
1220 FOR A=9 TO 20: GOSUB 1300: NEXT: RETURN
1300 FOR B=1 TO 3: C[A,B]=A[A,3,B]: NEXT: RETURN
1500 PLOT A[A,1],A[A,2]: DRAW A[C,1],A[C,2]: RETURN
1600 IF A[A,1]=A[C,1] THEN 1670 ELSE IF A[A,2]=A[C,2] THEN 1630
1610 P=(A[C,2]-A[A,2])/(A[C,1]-A[A,1])
1620 IF ABS(P)<1 THEN 1640 ELSE P=1/P: GOTO 1680
1630 P=0
1640 I=10*SGN(A[C,1]-A[A,1])/SQR(1+P*P)
1650 FOR Q=0 TO (A[C,1]-A[A,1])/I
1660 PLOT A[A,1]+Q*I,A[A,2]+Q*I*P: DRAWR 0.4*I,0.4*I*P: NEXT: RETURN
1670 P=0
1680 I=10*SGN(A[C,2]-A[A,2])/SQR(1+P*P)
1690 FOR Q=0 TO (A[C,2]-A[A,2])/I
1700 PLOT A[A,1]+Q*I*P,A[A,2]+Q*I: DRAWR 0.4*I*P,0.4*I: NEXT: RETURN
2000 TAG: MOVE 255,380: PRINT Z$;
2010 IF INKEY$="" THEN 2010
2020 TAGOFF: GOTO 20
```

N	C[20,3]	A[20,3,3]		
1	$1\,1\,1$	HKL	KLH	LHK
2	$1\,\bar{1}\,\bar{1}$	$H\bar{K}\bar{L}$	$K\bar{L}\bar{H}$	$L\bar{H}\bar{K}$
3	$\bar{1}\,\bar{1}\,1$	$\bar{H}\bar{K}L$	$\bar{K}\bar{L}H$	$\bar{L}\bar{H}K$
4	$\bar{1}\,1\,\bar{1}$	$\bar{H}K\bar{L}$	$\bar{K}L\bar{H}$	$\bar{L}H\bar{K}$
5	$\bar{1}\,\bar{1}\,\bar{1}$	$H\bar{K}\bar{L}$	$\bar{K}\bar{L}\bar{H}$	$\bar{L}H\bar{K}$
6	$\bar{1}\,1\,1$	HKL	$\bar{K}L\bar{H}$	$\bar{L}\bar{H}K$
7	$1\,1\,\bar{1}$	$\bar{H}K\bar{L}$	KLH	$L\bar{H}\bar{K}$
8	$1\,\bar{1}\,1$	$\bar{H}\bar{K}L$	$K\bar{L}\bar{H}$	LHK
9	$1\,0\,1$	$\bar{H}\bar{K}L$	HKL	LHK
10	$\bar{1}\,0\,1$	HKL	$\bar{H}\bar{K}L$	$\bar{L}\bar{H}K$
11	$1\,0\,\bar{1}$	$H\bar{K}\bar{L}$	$\bar{H}K\bar{L}$	$L\bar{H}\bar{K}$
12	$\bar{1}\,0\,\bar{1}$	$\bar{H}K\bar{L}$	$H\bar{K}\bar{L}$	$\bar{L}H\bar{K}$
13	$0\,1\,1$	$\bar{K}L\bar{H}$	KLH	HKL
14	$0\,1\,\bar{1}$	KLH	$\bar{K}L\bar{H}$	$\bar{H}K\bar{L}$
15	$0\,\bar{1}\,1$	$K\bar{L}\bar{H}$	$\bar{K}\bar{L}H$	$\bar{H}\bar{K}L$
16	$0\,\bar{1}\,\bar{1}$	$\bar{K}\bar{L}H$	$K\bar{L}\bar{H}$	$H\bar{K}\bar{L}$
17	$1\,1\,0$	$L\bar{H}\bar{K}$	LHK	KLH
18	$1\,\bar{1}\,0$	LHK	$L\bar{H}\bar{K}$	$K\bar{L}\bar{H}$
19	$\bar{1}\,1\,0$	$\bar{L}\bar{H}K$	$\bar{L}H\bar{K}$	$\bar{K}L\bar{H}$
20	$\bar{1}\,\bar{1}\,0$	$\bar{L}H\bar{K}$	$\bar{L}\bar{H}K$	$\bar{K}\bar{L}H$

Cristal cúbico de clase 23
Matriz de control y matriz de triplete de cara

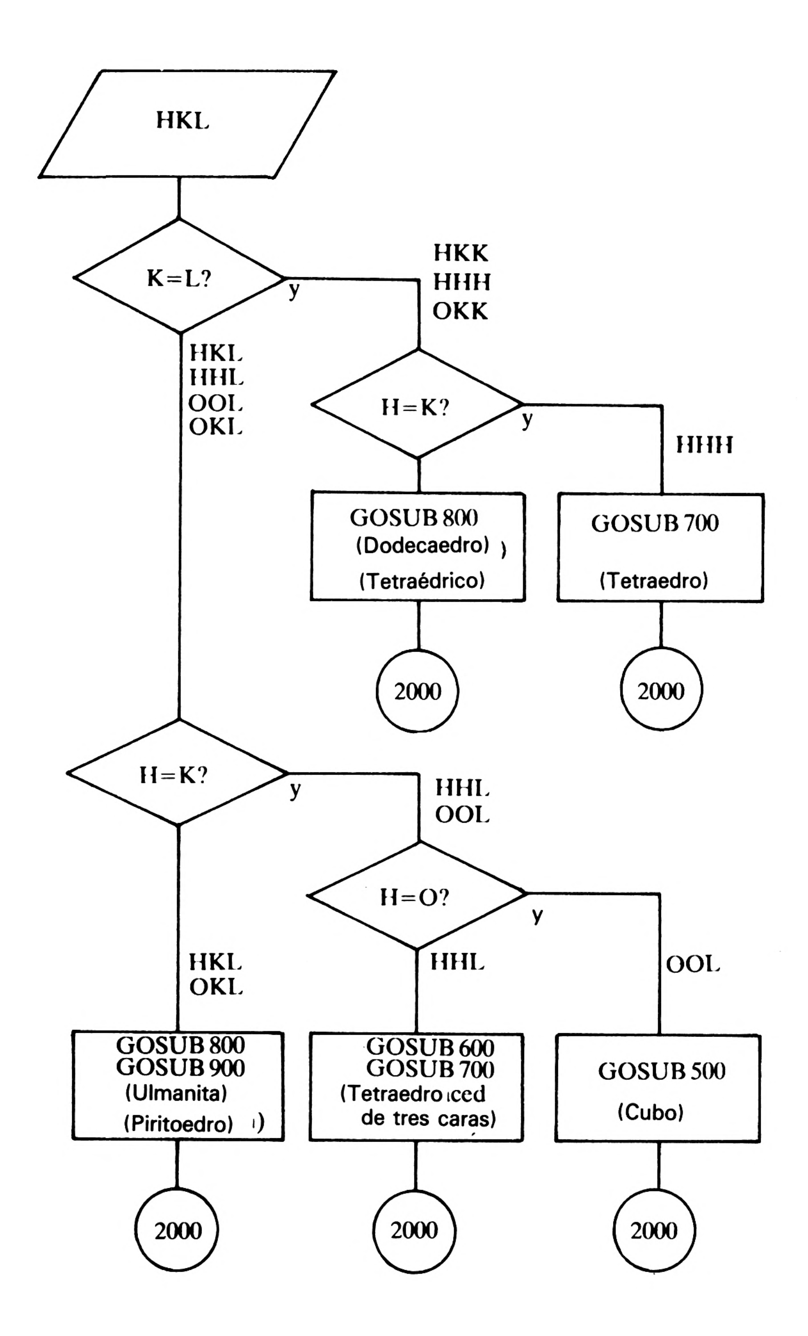


Fig. 4.1. Organigrama de separación de caras del cristal cúbico de clase 23

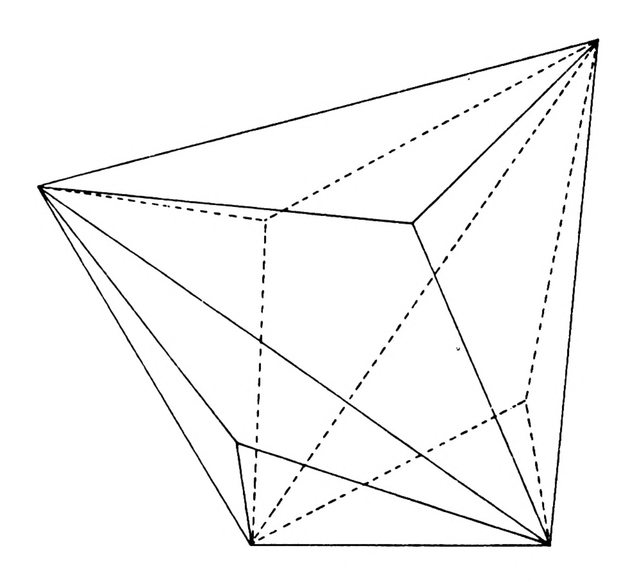


Fig. 4.2. Cristales cúbicos de clase 23. Forma {211}. Tetraedro de tres caras.

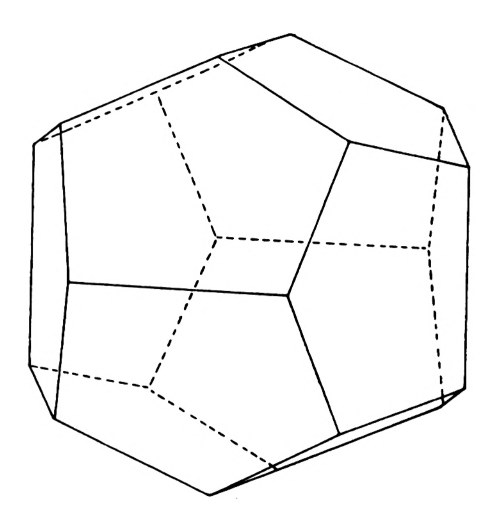


Fig. 4.3. Cristales cúbicos de clase 23. Forma {210}. Piritoedro.

5. Evaluación de un determinante por el desarrollo de Laplace

Un determinante es una abreviatura matemática para la suma y diferencia de un conjunto de productos.

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} * a_{22} - a^{21} * a_{12}$$

Se puede obtener determinantes más complicados por expansión, de la forma siguiente:

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} * \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} * \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} * \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

etc.

Como el número de términos viene dado por el factorial de N, el tamaño del resultado impreso se incrementa rápidamente. Los diferentes términos del producto contienen todas las permutaciones de los términos del determinante, y las líneas 190-290 son la rutina Permutación. Sin embargo, el orden en que se generan los términos requiere que se cambie el signo de cada término que sea factorial de los números impares (es decir, 1, 6, 120, 5040, etc.). Las líneas 120-160 averiguan los factoriales, y las líneas 250 y 260 comprueban si hay que cambiar el signo.

Los términos individuales y su valor se obtienen en las líneas 220-240, pero el programa es un buen ejemplo de cómo no se debe resolver un problema matemático escogiendo el método 'obvio'. En términos de tiempo de ejecución, el método de la matriz de inversión, explicado en otro apartado de este libro, es mucho más eficaz. Sin embargo, como la rutina siguiente sólo contiene productos y sumas, funcionará siempre, no importa lo mal condicionado que esté el determinante.

La función definida en la línea 40 sirve para darle a la impresión el formato empaquetado. NL$ es una operación de línea nueva.

```
10 REM Evaluacion de un determinante por el desarrollo de Laplace
20 DEFINT B,C,M,N,P,Q,S,T,X,Y,Z
30 DEF FN P$(P)=MID$(STR$(P),2): NL$=CHR$(10)+CHR$(13)
40 SYMBOL AFTER 208: H=HIMEM+1: POKE H,0
50 CLS: GOSUB 400:
60 LINE INPUT "Introduzca el orden del determinante  " ;A$: GOSUB 340: IF
Z=1 THEN Z=0: GOTO 60
70 DIM D[N,N]: PRINT: FOR P=1 TO N: FOR Q=1 TO N
80 PRINT "a"+FN P$(P)+FN P$(Q);"=";: INPUT D[P,Q]: NEXT Q,P
90 IF N=1 THEN CLS: GOSUB 400: PRINT "El determinante es a11=";D[1,1]:
```

```
GOTO 390
100 M=N-1: Y=1+INT(M/2)
110 DIM A[M],B[M],F[N],G[Y]
120 B=3: D$="": D=0: S=0: T=1
130 FOR P=1 TO N: F[P]=P: NEXT P
140 FOR A=1 TO Y: G[A]=2
150 FOR C=3 TO B: G[A]=G[A]*C: NEXT C
160 B=B+2: NEXT A
170 CLS: GOSUB 400
180 PRINT "El determinante es";NL$
190 FOR P=1 TO M: A[P]=1: B[P]=N-P+1: NEXT P
200 FOR P=1 TO M: IF A[P]>B[P] THEN A[P]=1
210 NEXT P: A=1: A$=""
220 FOR P=1 TO N: A=A*D[P,F[P]]: A$=A$+"a"+FN P$(P)+FN P$(F[P]):
NEXT P
230 D=D+T*A: D$=CHR$(44-T)+A$: S=S+1: T=-T
240 PRINT D$;
250 IF N>3 THEN GOSUB 300
260 FOR X=M TO 1 STEP -1
270 Z=F[X]: FOR P=X TO M: F[P]=F[P+1]: NEXT P: F[N]=Z
280 A[X]=A[X]+1: IF A[X]>B[X] THEN NEXT X: GOTO 380
290 GOTO 200
300 FOR P=1 TO Y: IF INT(S/G[P])=S/G[P] THEN T=-T
310 NEXT P: RETURN
320 IF A$="" OR VAL(A$)=0 THEN Z=1
330 RETURN
340 T=1: IF LEFT$(N$,1)="+" THEN T=2
350 FOR P=T TO LEN(A$): IF MID$(A$,T,1)<>"0" AND MID$(A$,P,1)>="0"
AND MID$(A$,P,1)<="9" THEN 370
360 Z=1: RETURN
370 NEXT P: N=VAL(A$): RETURN
380 PRINT: PRINT : PRINT "Det D=";D
390 END
400 LOCATE 6,2: PRINT "PRESENTACION Y VALOR DEL
DETERMINANTE";NL$;NL$
410 LOCATE 6,3: FOR P=1 TO 29: PRINT CHR$(208);: NEXT
420 PRINT: RETURN
```

Ejemplo

```
            PRESENTACION Y VALOR DEL DETERMINANTE

Introduzca el orden del determinante? 4
D(1,1)=? -2
D(1,2)=?  4
```

```
D(1,3)=? 7
D(1,4)=? 3
D(2,1)=? 8
D(2,2)=? 2
D(2,3)=? -9
D(2,4)=? 5
D(3,1)=? -4
D(3,2)=? 6
D(3,3)=? 8
D(3,4)=? 4
D(4,1)=? 2
D(4,2)=? -9
D(4,3)=? 3
D(4,4)=? 8

El determinante es
+a11a22a33a44 -a11a22a34a43 +a11a23a34a42
-a11a23a32a44 +a11a24a32a43 -a11a24a33a42
-a12a23a34a41 +a12a23a31a44 -a12a24a31a43
+a12a24a33a41 -a12a21a33a44 +a12a21a34a43
+a13a24a31a42 -a13a24a32a41 +a13a21a32a44
-a13a21a34a42 +a13a22a34a41 -a13a22a31a44
-a14a21a32a43 +a14a21a33a42 -a14a22a33a41
+a14a22a31a43 -a14a23a31a42 +a14a23a32a41

Det D= 2140
```

6. Triángulo

Este programa usa las relaciones geométricas de un triángulo para calcular los lados desconocidos y los ángulos dado que usted tiene tres sucesos conocidos, incluyendo la longitud de uno de los lados. Para comprobar que el triángulo concuerda con sus datos, el programa trazará, si usted así lo desea, un dibujo a escala y etiquetado. Esto se concebió para ayudar en los dibujos de ingeniería.

Las líneas 30-80 son las reglas del seno y el coseno, siendo FN A y FN C el arco coseno y el arco seno respectivamente.

IN$ se divide para las sentencias INPUT 510, 610, 710, etc.

NL$ es una línea nueva usada en las sentencias PRINT.

Las líneas 190-310 son el menú.

Las líneas 340-370 y 400-460 dan la presentación completa de los datos de PRINT. (T=3 representa el caso ambiguo en el que son posibles dos respuestas.)

Las subrutinas 500, 600, 700, 800 y 900 son las rutinas INPUT.

SUB 2000 es la cabecera y el subrayado.

SUB 2040 contiene una instrución para seguir adelante.

SUB 2070 es un filtro de números para 1-5 del menú.

SUB 2100 averigua si está listo para continuar.

Las subrutinas 2120, 2160, 2180, 2200 y 2220 son comprobaciones que aseguran que los datos introducidos pueden formar un triángulo, por ejemplo, la suma de los dos lados más cortos debe ser mayor que el tercer lado.

SUB 2240 es la sentencia 'Datos no válidos' con FAIL como portadora. El resto del programa dibuja un triángulo a escala. Excepto para el caso de que T=3 y VERT>320, el lado mayor se dibuja siempre como una línea horizontal de longitud fija, S. Observe el uso de 0.91 para la relación de aspecto de la pantalla con el fin de obtener la figura correcta.

Las líneas 2270-2290 encuentran el lado mayor, y, después, las líneas 2300-2350 reetiquetan los lados y las esquinas.

Las líneas 2390-2400 dibujan los otros dos lados y, si T=3, la línea 2410 dibuja la línea adicional para conseguir dos triángulos.

Las líneas restantes, 2410-2570, etiquetan los lados y los ángulos en el lugar correcto de la pantalla usando la subrutina 'impresión con TAG', representada por SUB 2700.

La subrutina vuelve a la línea 490 y después a la 180 para seguir con el triángulo siguiente después de la instrucción 'Pulse una tecla para continuar'.

```
10 REM Triangulo
20 DEG: DIM T$[5]: SYMBOL AFTER 208: HM=HIMEM+1: POKE HM,0
30 DEF FN Z(A,B,C)=(B*B+C*C-A*A)/2/B/C
40 DEF FN A(A)=90-ATN(A/SQR(1-A*A))
50 DEF FN B(A,Y,C)=SQR(A*A+C*C-2*A*C*COS(Y))
60 DEF FN X(A,Y,B)=A*SIN(Y)/B
70 DEF FN C(C)=ATN(C/SQR(1-C*C))
80 DEF FN D(A,Y,X)=A*SIN(Y)/SIN(X)
90 DEF FN U(U$)=-A*(U$="A")-B*(U$="B")-C*(U$="C")
100 DEF FN V(U$)=-X*(U$="X")-Y*(U$="Y")-Z*(U$="Z")
110 IN$="Introduzca a, b y c y los angulos X, Y y Z (en grados)"
120 NL$=CHR$(10)+CHR$(13): NL$=NL$+NL$: D$=" Grados"
130 CLS: GOSUB 2000
140 PRINT " Los triangulos quedan definidos si se conocen tres de los seis caracte-
res (tres lados mas tres angulos), siempre que al menos se de un lado."
150 PRINT " Hay un caso -dos lados y un angulo- en el que pueden formarse dos
triangulos si el angulo no es el que se ha incluido."
160 PRINT "      Este programa encuentra los valores de los angulos y lados desco-
nocidos, y dibuja el triangulo"
170 GOSUB 2040
180 CLS: GOSUB 2000
```

```
190 PRINT "     Elija la combinacion de la informacion conocida"
200 T$[1]="     1. Tres lados"
210 T$[2]="     2. Dos lados y el angulo incluido"
220 T$[3]="     3. Dos lados y el angulo no incluido"
230 T$[4]="     4. Un lado y dos angulos (ambos adyacentes)"
240 T$[5]="     5. Un lado y dos angulos (uno adyacente)"
250 FOR P=1 TO 5: LOCATE 1,3*P+4: PRINT T$[P]: NEXT P
260 LOCATE 1,22: LINE INPUT "Escriba 1, 2, 3, 4, o 5 ";T$[0]
270 GOSUB 2070
280 IF FAIL THEN 180
290 T=VAL(T$[0])
300 ON T GOSUB 500,600,700,800,900
310 S=370
320 IF T=3 THEN 390
330 CLS: GOSUB 2000: LOCATE 1,4
340 PRINT "a=";A;M$;NL$;"b=";B;M$;NL$;"c=";C;M$;NL$;"X=";USING
"###.####";X;
350 PRINT D$;NL$;"Y=";USING "###.####";Y;
360 PRINT D$;NL$;"Z=";USING "###.####";Z;
370 PRINT D$
380 C2=0: VERT=0: GOTO 490
390 CLS: GOSUB 2000: LOCATE 1,4
400 PRINT "a=";A;M$;NL$;"b=";B;M$;NL$;"c1="C1;M4;NL$;"c2=";C2;M$
410 PRINT: PRINT "X1=";USING "###.####";X1;
420 PRINT D$;NL$;"X2=";USING "###.####";X2;
430 PRINT D$;NL$;"Y=";USING "###.####";Y;
440 PRINT D$;NL$;"Z1="USING "###.####";Z1;
450 PRINT D$;NL$;"Z2="USING "###.####";Z2;
460 PRINT D$
470 C=C1+C2*(SGN(C2)=-1): X=X1: Z=Z1: VERT=A*SIN(Y)*370/C
480 SW=A: A=B: B=SW: SW=X: X=Y: Y=SW: IF VERT>320 THEN
S=S*320/VERT
490 GOSUB 2040: GOSUB 2260: GOSUB 2040: GOTO 180
500 CLS: GOSUB 2000
510 PRINT: PRINT LEFT$(IN$,16)
520 GOSUB 1000: GOSUB 1010: GOSUB 1020: GOSUB 1060
530 GOSUB 2100: IF FAIL THEN 500
540 H=A: K=B: L=C: GOSUB 2120: IF FAIL THEN 500
550 X=FN A(FN Z(A,B,C)): Y=FN A(FN Z(B,C,A)): Z=FN A(FN Z(C,A,B))
560 RETURN
600 CLS: GOSUB 2000
610 PRINT: PRINT LEFT$(IN$,8)+MID$(IN$,15,7)+MID$(T$[2],22,12)+RIGHT$
(T$[2],6)+" Y "+RIGHT$(IN$,12)
620 GOSUB 1000: GOSUB 1020: GOSUB 1040: GOSUB 1060
630 GOSUB 2100: IF FAIL THEN 600
640 GOSUB 2160: IF FAIL THEN 600
```

```
650 B=FN B(A,Y,C): X=FN C(FN X(A,Y,B))
660 IF A*COS(Y)>C THEN X=180-X
670 Z=180-X-Y
680 RETURN
700 CLS: GOSUB 2000
710 PRINT: PRINT LEFT$(IN$,10)+MID$(IN$,11,5)+MID$(T$[3],22,16)+RIGHT$
(T$[3],6)+" Y "+RIGHT$(IN$,12)
720 GOSUB 1000: GOSUB 1010: GOSUB 1040: GOSUB 1060
730 GOSUB 2100: IF FAIL THEN 700
740 GOSUB 2180: IF FAIL THEN 700
750 X1=FN C(FN X(A,Y,B)): Z1=180-X1-Y: Z2=X1-Y
760 X2=180-X1: C1=FN D(B,Z1,Y): C2=FN D(B,Z2,Y)
770 RETURN
800 CLS: GOSUB 2000
810 PRINT: PRINT LEFT$(IN$,7)+MID$(IN$,17,16)+MID$(IN$,36)
820 GOSUB 1000: GOSUB 1040: GOSUB 1050: GOSUB 1060
830 GOSUB 2100: IF FAIL THEN 800
840 GOSUB 2200: IF FAIL THEN 800
850 X=180-Y-Z: B=FN D(A,Y,X): C=FN D(A,Z,X)
860 RETURN
900 CLS: GOSUB 2000
910 PRINT: PRINT LEFT$(IN$,7)+MID$(IN$,17,17)+RIGHT$(IN$,19)
920 GOSUB 1000: GOSUB 1030: GOSUB 1050: GOSUB 1060
930 GOSUB 2100: IF FAIL THEN 900
940 GOSUB 2220: IF FAIL THEN 900
950 Y=180-X-Z: B=FN D(A,Y,X): C=FN D(A,Z,X)
960 RETURN
1000 LOCATE 5,8: INPUT "a=";A: RETURN
1010 LOCATE 5,10: INPUT "b=";B: RETURN
1020 LOCATE 5,12: INPUT "c=";C: RETURN
1030 LOCATE 5,14: INPUT "x=";X: RETURN
1040 LOCATE 5,16: INPUT "Y=";Y: RETURN
1050 LOCATE 5,18: INPUT "Z=";Z: RETURN
1060 LOCATE 5,20: INPUT "Unidad de longitud";M$: RETURN
2000 LOCATE 16,1: PRINT "TRIANGULO"
2010 LOCATE 16,2: FOR P=1 TO 9: PRINT CHR$(208);: NEXT P
2020 RETURN
2040 LOCATE 1,25: PRINT "Pulse cualquier tecla para continuar"
2050 IF INKEY$="" THEN 2050
2060 RETURN
2070 FAIL=0: IF T$[0]="" THEN FAIL=-1: RETURN
2080 IF T$[0]>="1" AND T$[0]<="5" THEN RETURN
2090 FAIL=-1: RETURN
2100 FAIL=0: LOCATE 1,23: PRINT "      Listo? (s/n) ";: INPUT K$:
K$=UPPER$(K$): IF K$<>"S" THEN FAIL=-1
2110 RETURN
```

```
2120 FAIL=0: IF H<K THEN SW=H: H=K: K=SW
2130 IF K<L THEN SW=K: K=L: L=SW: GOTO 2120
2140 IF H<K+L THEN RETURN
2150 GOSUB 2240: RETURN
2160 FAIL=0: IF Y<180 THEN RETURN
2170 GOSUB 2240: RETURN
2180 FAIL=0: IF B>A*SIN(Y) THEN RETURN
2190 GOSUB 2240: RETURN
2200 FAIL=0: IF Y+Z<180 THEN RETURN
2210 GOSUB 2240: RETURN
2220 FAIL=0: IF X>0 THEN RETURN
2230 GOSUB 2240: RETURN
2240 LOCATE 27,8: PRINT "Datos no validos": GOSUB 2040: FAIL=-1: RETURN
2260 CLS: PLOT 135,48: DRAWR S,0: IF VERT>320 THEN P=2: GOTO 2300
2270 IF A>=B AND A>=C THEN P=0
2280 IF B>=C AND B>=A THEN P=1
2290 IF C>=A AND C>=B THEN P=2
2300 H$=CHR$(65+P): H=FN U(H$)
2310 K$=CHR$(66+P+(P>1)*3): K=FN U(K$)
2320 L$=CHR$(67+P+(P>0)*3): L=FN U(L$)
2330 G$=CHR$(88+P): G=FN V(G$)
2340 I$=CHR$(89+P+(P>1)*3): I=FN V(I$)
2350 J$=CHR$(90+P+(P>0)*3): J=FN V(J$)
2360 SCY=S/H: SCX=SCY*0.91
2370 CJ=SCY*COS(J): CI=SCY*COS(I)
2380 SJ=SCX*SIN(J): SI=SCX*SIN(I)
2390 DRAWR -K*CJ,K*SJ
2400 DRAWR -L*CI,-L*SI
2410 IF T=3 THEN PLOT 135+S-K*CJ,48+K*SJ: DRAWR -K*CJ,-K*SJ
2420 E=260: F=44
2430 TEXT$=H$+"="+LEFT$(STR$(H),7): GOSUB 2700
2440 E=100+(SGN(C2)=-1)*(C2*S/C): F=56+12*(SGN(C2)=-1)
2450 TEXT$=I$+"=": GOSUB 2700
2460 E=80+(SGN(C2)=-1)*(C2*S/C-10): F=40+12*(SGN(C2)=-1)
2470 TEXT$=LEFT$(STR$(I),7): GOSUB 2700
2480 E=510: F=56
2490 TEXT$=J$+"=": GOSUB 2700
2500 E=500: F=40
2510 TEXT$=LEFT$(STR$(J),7): GOSUB 2700
2520 E=480-K*CJ: F=64+K*SJ
2530 TEXT$=G$+"="+LEFT$(STR$(G),7): GOSUB 2700
2540 E=550-0.5*K*CJ: F=56+0.5*K*SJ
2550 TEXT$=K$+"="+LEFT$(STR$(K),7): GOSUB 2700
2560 E=30+(SGN(C2)=-1)*2*C2*S/C+0.5*L*CI: F=56+0.5*L*SI
2570 TEXT$=L$+"="+LEFT$(STR$(L),7): GOSUB 2700
2580 E=260: F=390: TEXT$="TRIANGULO": GOSUB 2700
```

```
2590 PLOT 260,370: DRAWR 128,0
2600 RETURN
2700 FOR Q=1 TO LEN(TEXT$): Z$=MID$(TEXT$,Q,1)
2710 MOVE E,F: TAG
2720 PRINT Z$;
2730 E=E+16+16*(Z$=" "): NEXT Q
2740 TAGOFF: RETURN
```

Ejemplo

TRIANGULO

Introduzca a y c y el angulo incluido Y (en grados)

```
=? 25
c=? 37
y=? 45
```

Unidad de longitud? mm Listo? (y/n)? s

```
a= 25mm
b= 26.1887849mm
c=37mm
X= 42.4549 grados
Y= 45.0000 grados
Z= 92.5451 grados
```

Pulse cualquier tecla para continuar

7. Estructura de árbol 2-3

Un árbol 2-3 es una estructura de árbol en la que cada vértice distinto de cero tiene dos o tres ramas. La información almacenada se lleva a las hojas y, para encontrar la hoja apropiada, cada vértice que lleva de una hoja hacia la raíz contiene dos fragmentos adicionales de información. El diagrama facilita la comprensión de todo esto. Para encontrar una hoja en particular, usted hace dos preguntas en cada vértice, empezando en la raíz.

1. ¿Es el valor de la hoja menor o igual que el número de la izquierda? Si es así, baja a la rama izquierda.
2. Si no es así, ¿es el valor de la hoja menor o igual que el número de la derecha? De ser así, baja a la rama central. De lo contrario, baja a la rama derecha.

Para cumplir la regla 2 o 3, es fácil mostrar que el número máximo de hojas es $3 \uparrow N$ y el mínimo $2 \uparrow N$, donde N es la altura del árbol. La tabla siguiente muestra el margen de las capacidades para cada tamaño de árbol.

Altura del árbol	*Mínimo*	*Máximo*
0	1	1
1	2	3
2	4	9
3	8	27
4	16	81
5	32	243
6	64	729
7	128	2187

Por tanto, con un árbol 2-3 de altura 7 puede usted encontrar una hoja entre 2187 con un número de preguntas del tipo expuesto arriba que varía entre 7 y 13 (la media está en 11), más 8 preguntas como ésta: '¿Esto es una hoja?', lo cual hace un total de 19 preguntas.

Con un árbol binario simétrico que contenga 2047 elementos la altura mínima debe ser 11, y usted necesitaría entre 1 y 21 preguntas (por ejemplo, '¿Esto es un vértice? Si no es así, ¿es menor?'), estando la media en 19 preguntas también.

El problema que tiene el árbol binario es que la suma o la eliminación de miembros produce un árbol asimétrico, particularmente si las entradas se realizan en orden (véase 'Árbol de búsqueda binario'), y entonces el tiempo de acceso se hace significativamente diferente del mejor valor, obtenido en un árbol simétrico. El tiempo de acceso para un árbol 2-3 es constante y depende sólo del tamaño inicial del árbol.

El organigrama muestra la construcción general del programa, basado en menú de nueve opciones. Siguiendo este esquema, el programa averigua primero si ya se lo ha utilizado para establecer una base de datos y, si es así, pide que ésta sea introducida en el ordenador (SUB 300). De lo contrario, SUB 5000 forma un nuevo árbol. Esta subrutina establece el tamaño mínimo del árbol requerido (línea 5020) y después, utilizando el 'bucle anidado de tamaño variable', encuentra la localización de las posiciones de almacenamiento para cumplir con la regla 2 o 3. Las variables B1, B2 y B3 transforman los valores finales del bucle K de 0 a 1, de forma que en cada vértice el bucle vaya de -1 a 1 (-1 TO 1; tres ramas) o de -1 a 0 (-1 TO 0; dos ramas), como es preciso. Los valores de decisión del árbol se almacenan en las matrices L y M (izquierda y media, respectivamente). La última parte de la matriz L (que es mayor que la matriz M) contiene los datos que hay que almacenar. Éstos son introducidos (INPUT) en la línea 5150, ordenados mediante una clasificación de montones en SUB 1000 y cargados en L en la línea 5170.

A continuación, el programa se ejecuta bajo el control del menú (línea 200, a la cual debe volver usted en caso de necesidad). La mayor parte de las subrutinas son directas, con la excepción de 500, 2000 y 3000.

SUB 500 genera la matriz H, necesaria para la reorganización del árbol en 2000 y 3000. La matriz H contiene las posiciones de almacenamiento en el desbordamiento,

es decir, para $3\uparrow N+1$ valores cuando se los almacena en un espacio lo más reducido posible dentro de las reglas 2,3. La tabla da los 50 primeros valores.

SUB 2000 añade un nuevo valor al árbol existente. Antes que nada, encuentra el valor de V para el padre (SUB 1800); hay tres casos posibles:

1. El padre tiene dos hijos, en cuyo caso se puede añadir un tercero (SUB 2300).
2. El padre ya tiene tres hijos, en cuyo caso hay que encontrar un padre que sólo tenga dos hijos (bucles 2040 y 2070) en la posición A. Los hijos interpuestos son apartados con el fin de hacer sitio para la nueva entrada (líneas 2100-2150 para desplazamiento a la izquierda y líneas 2170-2220 para desplazamiento a la derecha).
3. No se puede encontrar un padre con sólo dos hijos. En este caso el árbol tiene que ser reorganizado en SUB 2500 para emplear algunas de las hojas y ramas vacantes, pero siguiendo aún la regla 2 o 3. Esto se hace desplazando las posiciones vacantes con entradas por la derecha en la línea 2630 usando la función S, que se deriva de la información rebosada en la matriz H.

Una vez realizada la reorganización, se vuelve a intentar las dos primeras opciones para obtener una posición apropiada. Esto falla sólo si el árbol está lleno (línea 2560).

SUB 3000 sigue un procedimiento similar para eliminar una entrada. Los tres casos son:

1. El padre tiene tres hijos (SUB 3300).
2. El padre sólo tiene dos hijos, se encuentra un padre con tres hijos y los hijos situados entre las dos posiciones son desplazados a la izquierda o a la derecha (líneas 3110-3220).
3. No hay ningún padre que tenga tres hijos, en cuyo caso se reorganiza el árbol (SUB 3500) y se emplea el caso 1 o 2, a no ser que haya menos de $2\uparrow N$ entradas (N es la altura del árbol).

Observe el uso de CHR$(34) para conseguir comas invertidas en 4430, aunque también podría haberse usado WRITE.

Se puede adaptar el programa para que almacene información literal o, si cada fragmento de información tiene un único número, éste puede ser almacenado y usado como puntero hacia la matriz principal de almacenamiento de cadenas. (Véase 'Etiqueta'.)

Las matrices D, L y M deben ser matrices literales, X se debe transformar en X$ y los diversos ceros empleados en las comparaciones deben convertirse en "".

```
9 REM Estructura de arbol 2-3
10 DEFINT A,B,F-K, N,P,Q,V,Z
20 DEF FN L(L)=INT(3↑L-0.99999991)
30 DEF FN K(K)=1+K+FN L(K)/2
40 DEF FN S(A,B)=H(A)-H(1+B)+1
50 DEF FN A$(A)=MID$(STR$(A),2)
```

```
60 NL$=CHR$(10)+CHR$(13): DIM M$[9]
70 M$[1]="Crear un arbol 2-3 nuevo"
80 M$[2]="Encontrar un miembro"
90 M$[3]="Encontrar min/max y numero de miembros"
100 M$[4]="Añadir un miembro nuevo"
110 M$[5]="Eliminar un miembro"
120 M$[6]="Imprimir una lista de miembros"
130 M$[7]="Grabar los datos en disco"
140 M$[8]="Grabar el programa"
150 M$[9]="Fin"
160 WINDOW #1,1,40,23,25: CLS: GOSUB 1600
170 PRINT "  Este programa esta diseñado para crear una estructura de
almacenamiento de arbol 2-3, usando una unidad de disco para la grabacion y
recuperacion de los datos."
180 PRINT "  Si ya ha creado y grabado el arbol, pulse"+CHR$(34)+"S  e
<INTRO>"+CHR$(34)+"; de lo contrario pulse solo "+CHR$(34)+"<INTRO>"
+CHR$(34)
190 LINE INPUT K$: K$=UPPER$(K$): IF K$="S" THEN GOSUB 300 ELSE
GOSUB 5000: GOSUB 500
200 CLS: GOSUB 1600
210 FOR P=1 TO 9: LOCATE 2,2+2*P: PRINT P;CHR$(8);". ";M$[[P]: NEXT
220 LOCATE #1,1,1: PRINT #1,"Escriba su eleccion: 1-9";
230 LOCATE 25,23: INPUT M$[0]: GOSUB 1650
240 IF FAIL THEN 200
250 K=VAL(M$[0])
260 ON K GOSUB 4700,4500,4000,2000,3000,4600,4100,4300,4400
270 GOTO 200
299 REM Cargar datos del disco
300 CLS: GOSUB 1600: PRINT "Esta preparada la unidad de disco para introducir
los datos almacenados (S/N)?";NL$
310 IF INKEY$<>"S" AND INKEY$<>"s" THEN 310
320 INPUT "Nombre del fichero de datos";TREE$
330 OPENIN TREE$
340 INPUT #9,N,M: AL=FN L(N+1)/2: AM=FN L(N)/2: DIM L[AL],M[AM]
350 FOR P=1 TO AL: INPUT #9,L[P]: NEXT: INPUT #9,SMALL
360 FOR P=1 TO AM: INPUT #9,M[P]: NEXT: INPUT #9,LARGE
370 CLOSEIN
380 GOSUB 500
390 RETURN
499 REM Derivacion de la matriz H
500 NH=FN K(N): DIM H[NH]: H[1]=1
510 I=0: J=0: K=1: Y=1: Z=-1
520 FOR P=1 TO FN K(N-1)
530 IF K[K]=9 THEN I=1: Y=7: J=0: Z=1
540 FOR Q=-1 TO -(I>0)
550 IF H[P]<>0 THEN H[K+1]=3*H[P]+Q: K=K+1: J=J+1
```

```
560 NEXT Q
570 IF J<3*FN L(Z) THEN 600
580 I=0
590 NZ=FN L(Z): IF P>3 THEN IF H[Y+3*NZ+4]/H[Y]=3 THEN I=1: J=0:
Y=Y+3*NZ+4: Z=Z+1
600 NEXT P
610 RETURN
999 REM Clasificacion por montones
1000 FOR P=INT(M/2) TO 1 STEP -1: R=P
1010 S=R+R: T=S+1
1020 IF D[R]<D[S] THEN GOSUB 1170: GOTO 1050
1030 IF T<=M THEN IF D[R]<D[T] THEN GOSUB 1200: GOTO 1050
1040 GOTO 1060
1050 IF R<=M/2 THEN 1010
1060 NEXT P
1070 FOR P=M TO 3 STEP -1: SW=D[1]: D[1]=D[P]: D[P]=SW: R=1
1080 S=R+R: T=S+1
1090 IF D[R]<D[S] THEN GOSUB 1150: GOTO 1120
1100 IF T<P THEN IF D[R]<D[T] THEN GOSUB 1200: GOTO 1120
1110 GOTO 1130
1120 IF R<=(P-1)/2 THEN 1080
1130 NEXT P
1140 SW=D[1]: D[1]=D[2]: D[2]=SW: RETURN
1150 IF T>=P THEN 1190
1160 GOTO 1180
1170 IF T>M THEN 1190
1180 IF D[S]<D[T] THEN 1200
1190 SW=D[R]: D[R]=D[S]: D[S]=SW: R=R+R: RETURN
1200 SW=D[R]: D[R]=D[T]: D[T]=SW: R=R+R+1: RETURN
1499 REM Numero
1500 FOR H=N-1 TO 0 STEP -1
1510 FOR V=FN L(H)/2+1 TO FN L(H+1)/2
1520 T=H: VL=3*V-1: VM=VL+1
1530 IF T=N-1 THEN 1570
1540 VL=3*VL-(L[3*VL+1]>0)
1550 VM=3*VM-(L[3*VM+1]>0)
1560 T=T+1: GOTO 1530
1570 L[V]=L[VL]: M[V]=L[VM]
1580 NEXT V,H
1590 RETURN
1599 REM Titulo
1600 LOCATE 11,2: PRINT "ESTRUCTURA DE ARBOL 2-3": LOCATE 11,3
1610 FOR P=1 TO 23: PRINT CHR$(208);: NEXT: PRINT
1620 RETURN
1649 REM Prueba de M$[0]
1650 FAIL=0: IF M$[0]="" THEN 1670
```

```
1660 IF LEN(M$[0])=1 AND M$[0]>="1" AND M$[0]<="9" THEN RETURN
1670 FAIL=-1: RETURN
1699 REM Llevar la cuenta de los miembros
1700 M=M+1: SMALL=0
1710 IF M=3↑N THEN PRINT "Estructura del arbol llena": LARGE =-1:
GOSUB 4750
1720 RETURN
1730 M=M-1: LARGE=0
1740 IF M=2↑N THEN PRINT "Arbol de tamaño minimo": SMALL=-1
GOSUB 4750
1750 RETURN
1799 REM Comprobacion de los miembros
1800 V=1: H=1: MEMBER=0
1810 IF H<N THEN GOSUB 1850: H=H+1: GOTO 1810
1820 V3=3*V
1830 IF X=L[V3-1] OR X=L[V3] OR X=L[V3+1] THEN MEMBER=-1
1840 RETURN
1850 IF X<=L[V] THEN V=3*V-1: RETURN
1860 IF X<=M[V] THEN V=3*V: RETURN
1870 V=3*V+1: RETURN
1999 REM A«adir hijo
2000 CLS: GOSUB 1600: INPUT "Miembro nuevo";X: IF X<=0 THEN 2000
2005 IF LARGE=-1 THEN PRINT "Arbol lleno": GOSUB 4750: RETURN
2010 GOSUB 1800: IF MEMBER THEN PRINT "Ya hay un miembro": GOSUB 4750:
RETURN
2020 IF L[V]<>0 AND L[3*V+1]=0 THEN GOSUB 2300: GOSUB 1500:
GOTO 1700
2030 GOSUB 2350
2040 FOR P=FN L(N-1)/2+1 TO V-1
2050 IF L[P]<>0 THEN IF L[3*P+1]=0 THEN A=P: GOTO 2100
2060 NEXT P
2070 Z=Z+1: IF V+1>FN L(N)/2 THEN 2500 ELSE FOR P=V+1 TO FN L(N)/2
2080 IF L[P]<>0 THEN IF L[3*P+1]=0 THEN A=P: GOTO 2170
2090 NEXT P: GOTO 2500
2100 A=3*A+1: IF A=Z THEN L[A]=X: GOTO 2230
2110 B=A+1
2120 IF B<Z THEN IF L[B]=0 THEN B=B+1: GOTO 2120
2130 L[A]=L[B]: IF L[B]<>0 THEN A=B
2140 IF B<Z THEN GOTO 2110
2150 L[A]=X
2160 GOSUB 1500: GOTO 1700
2170 A=3*A+1
2180 B=A-1
2190 IF B>Z THEN IF L[B]=0 THEN B=B-1: GOTO 2190
2200 L[A]=L[B]: IF B>=Z THEN A=B
2210 IF B>Z THEN 2180
```

```
2220 L[B]=X
2230 GOSUB 1500: GOTO 1700
2299 REM Caso en el que el padre tiene solo dos hijos
2300 V3=V*3
2310 IF X<=L[V] THEN L[V3+1]=L[V3]: L[V3]=L[V3-1]: L[V3-1]=X: RETURN
2320 IF X<=M[V] THEN L[V3+1]=L[V3]: L[V3]=X: RETURN
2330 L[V3+1]=X: RETURN
2349 REM Busqueda de Z
2350 V3=V*3: IF L[V]=0 THEN Z=V3-1: RETURN
2360 IF X<L[V3-1] THEN Z=V3-2: RETURN
2370 IF X<L[V3] THEN Z=V3-1: RETURN
2380 IF X<L[V3+1] THEN Z=V3: RETURN
2390 Z=V3+1: RETURN
2499 REM Reorganizacion del arbol
2500 FOR H=N-2 TO 0 STEP -1
2510 FOR P=FN L(H)/2+1 TO FN L(H+1)/2
2520 V=P
2530 IF L[V]=0 THEN 2550
2540 IF L[3*V+1]=0 THEN GOSUB 2600: GOSUB 1500: GOSUB 1800:
GOTO 2020
2550 NEXT P
2560 NEXT H: PRINT "Si añadiese este miembro el arbol quedaria fuera de la
memoria.": GOSUB 4750: LARGE=-1: RETURN
2600 A=N-H-1: A1=2*(FN L(A)+1): SA=FN K(A)
2610 IF H<N THEN V=3*V+1: H=H+1: GOTO 2610
2620 V1=V-A1: V1K=V1-SA-1: FOR P=FN K(A+1) TO FN K(A)+1 STEP -1:
V1S=V1+FN S(P,SA):
2630 SW=L[V1S]: L[V1S]=L[V1K+P]: L[V1K+P]=SW
2640 NEXT P: RETURN
2999 REM Eliminar hijo
3000 CLS: GOSUB 1600: INPUT "Que entrada le gustaria eliminar";X: IF X<=0
THEN 3000
3010 IF SMALL=-1 THEN PRINT "Arbol de tamaño minimo": GOSUB 4750:
RETURN
3020 GOSUB 1800: IF NOT MEMBER THEN PRINT "No hay un miembro":
GOSUB 4750: RETURN
3030 IF L[3*V+1]<>0 THEN GOSUB 3300: GOSUB 1500: GOTO 1730
3040 GOSUB 3450
3050 FOR P=FN L(N-1)/2+1 TO V-1
3060 IF L[P]<>0 AND L[3*P+1]<>0 THEN A=P: GOTO 3110
3070 NEXT P
3080 IF V+1>FN L(N)/2 THEN 3500 ELSE FOR P=V+1 TO FN L(N)/2
3090 IF L[P]<>0 THEN IF L[3*P+1]<>0 THEN A=P: GOTO 3170
3100 NEXT P: GOTO 3500
3110 A=3*A+1
3120 B=Z-1
```

```
3130 IF B>A THEN IF L[B]=0 THEN B=B-1: GOTO 3130
3140 L[Z]=L[B]: Z=B
3150 IF B>A THEN 3120
3160 L[B]=0: GOSUB 1500: GOTO 1730
3170 A=3*A+1
3180 B=Z+1
3190 IF B<A THEN IF L[B]=0 THEN B=B+1: GOTO 3190
3200 L[Z]=L[B]: Z=B
3210 IF B<A THEN 3180
3220 L[B]=0: GOSUB 1500: GOTO 1730
3299 REM Caso en el que el padre tiene tres hijos
3300 IF X=L[V] THEN L[V3-1]=L[V3]: L[V3]=L[V3+1]: L[V3+1]=0: RETURN
3310 IF X=M[V] THEN L[V3]=L[V3+1]: L[V3+1]=0: RETURN
3320 L[V3+1]=0: RETURN
3449 REM Encontrar Z
3450 IF X=L[V] THEN Z=V3-1: RETURN
3460 Z=V3: RETURN
3499 REM Reorganizacion del arbol
3500 FOR H=N-2 TO 0 STEP -1
3510 FOR P=FN L(H)/2+1 TO FN L(H+1)/2
3520 V=P
3530 IF L[V]=0 THEN 3550
3540 IF L[3*V+1]<>0 THEN GOSUB 3570: GOSUB 1500: GOSUB 1800:
GOTO 3030
3550 NEXT P,H
3560 PRINT "Si eliminase este miembro, quedarian muy pocos elementos para el
tamaño de este arbol.": GOSUB 4750: SMALL=-1: RETURN
3570 A=N-H-1: A1=2*(FN L(A)+1): SA=FN K(A)
3580 IF H<N THEN V=3*V+1: H=H+1: GOTO 3580
3590 V1=V-A1: V1K=V1-SA-1: FOR P=FN K(A)+1 TO FN K(A+1):
V1S=V1+FN S(P,SA)
3600 SW=L[V1S]: L[V1S]=L[V1K+P]: L[V1K+P]=SW
3610 NEXT P
3620 RETURN
3999 REM Numero de miembros
4000 CLS: GOSUB 1600
4010 PRINT "El numero de miembros es";M;NL$
4020 SM=FN L(N): PRINT "El miembro menor es";L[SM/2+1];NL$
4030 V=1: H=0
4040 WHILE H<N: V=3*V-(L[3*V+1]<>0): H=H+1: WEND
4050 PRINT "El miembro mayor es";L(V)
4060 GOSUB 4750: RETURN
4099 REM Grabar los datos
4100 CLS: GOSUB 1600
4110 PRINT "Esta la unidad de disco preparada para grabar los datos (S/N)?";NL$
4120 IF INKEY$<>"S" AND INKEY$<>"s" THEN 4120
```

```
4130 INPUT "Nombre del fichero de datos";TREE$
4140 OPENOUT TREE$
4150 PRINT #9,N,M
4160 FOR P=1 TO FN L(N+1)/2: PRINT #9,L[P]: NEXT: PRINT #9,SMALL
4170 FOR P=1 TO FN L(N)/2: PRINT #9,M[P]: NEXT: PRINT #9,LARGE
4180 CLOSEOUT: PRINT "OK": GOSUB 4750
4190 RETURN
4299 REM Grabar programa
4300 CLS: GOSUB 1600
4310 PRINT "Esta la unidad de disco preparada para grabar el programa
(S/N)?";NL$
4320 IF INKEY$<>"S" AND INKEY$<>"s" THEN 4320
4330 SAVE "23TREE"
4340 PRINT "El programa grabado en el disco ya esta incluido en el directorio."
4370 CAT
4380 GOSUB 4750
4390 RETURN
4399 REM Fin
4400 CLS: GOSUB 1600
4410 PRINT "De verdad quiere acabar?";NL$
4420 PRINT "Ha grabado los datos?";NL$
4430 PRINT "Pulse ";CHR$(34);"S";CHR$(34);" para acabar; de lo contrario pulse
<INTRO>";: INPUT K$
4440 K$=UPPER$(K$): IF K$="S" THEN NEW ELSE RETURN
4499 REM Miembro?
4500 CLS: GOSUB 1600
4510 INPUT "Que miembro quiere comprobar";X
4520 GOSUB 1800
4530 IF MEMBER THEN PRINT NL$;X;" es un miembro" ELSE PRINT NL$;X;"
no es un miembro"
4540 GOSUB 4750: RETURN
4599 REM Lista de miembros
4600 CLS: GOSUB 1600
4610 PRINT "Lista de los miembros:": PRINT
4620 FOR P=FN L(N)/2+1 TO FN L(N+1)/2
4630 PRINT L[P];
4640 NEXT P: PRINT
4650 GOSUB 4750: RETURN
4699 REM Borrar matrices
4700 ERASE H,L,M
4710 GOSUB 5000: GOSUB 500
4720 RETURN
4749 REM Seguir
4750 PRINT #1, "Pulse cualquier tecla para continuar"
4760 IF INKEY$="" THEN 4760
4770 RETURN
```

```
4999 REM Arbol 2-3 nuevo
5000 CLS: GOSUB 1600: INPUT "Numero de valores que hay que almacenar"; M:
IF M<2 THEN 5000
5010 Y=M+1: N=-1
5020 WHILE Y>1: Y=Y/2: N=N+1: WEND
5030 V=1: Z=1: F=2↑N: G=3*2↑(N-1): NV=FN L(N+1)/2: NM=FN L(N)/2
5040 B1=3*(F-M)*(M<=G): B2=3*(G-M)*(M>G): B3=9*(F-M)*(M>G)
5050 DIM T[3^N],A[N],E[N],D[M],L[NV:],M[NM]
5060 FOR P=1 TO N: A[P]=-1: B[P]=0: NEXT P
5070 FOR P=1 TO N: IF A[P]>B[P] THEN A[P]=-1
5080 NEXT P
5090 B[N]=-(Z<B1)-(Z<B2): B[N-1]=-(Z<B3)
5100 FOR P=1 TO N: V=3*V+A[P]: NEXT P: T[Z]=V: Z=Z-1
5110 FOR K=N TO 1 STEP -1: V=1
5120 A[K]=A[K]+1: IF A[K]>B[K] THEN NEXT K ELSE 5070
5140 CLS: GOSUB 1600
5150 FOR P=1 TO M: PRINT *D(";FN A$(P);")=*;: INPUT D[P]: IF D[P]=0 THEN
P=P-1
5160 NEXT P
5170 GOSUB 1000
5180 FOR P=1 TO M: L[T[P]]=D[P]: NEXT P
5190 SMALL=0: LARGE=0
5200 IF M=2↑N THEN SMALL=-1
5210 IF M=3↑N THEN LARGE=-1
5220 ERASE A,B,D,T
5230 GOSUB 1500
5240 RETURN
```

El programa anterior está escrito específicamente para el CPC-6128. Los usuarios del CPC-464 deberán reemplazar las líneas 299-390, 4099-4180 y 4299-4380 con las órdenes propias del manejo de cintas.

La matriz H

N	H[N]	N	H[N]	N	H[N]
1	1	18	41	35	58
2	2	19	42	36	68
3	3	20	43	37	69
4	5	21	44	38	70
5	6	22	45	39	71
6	8	23	46	40	72
7	9	24	47	41	73
8	14	25	48	42	77
9	15	26	49	43	78

N	H[N]	N	H[N]	N	H[N]
10	16	27	50	44	80
11	17	28	51	45	81
12	18	29	52	46	122
13	19	30	53	47	123
14	23	31	54	48	124
15	24	32	55	49	125
16	26	33	56	50	126
17	27	34	57		

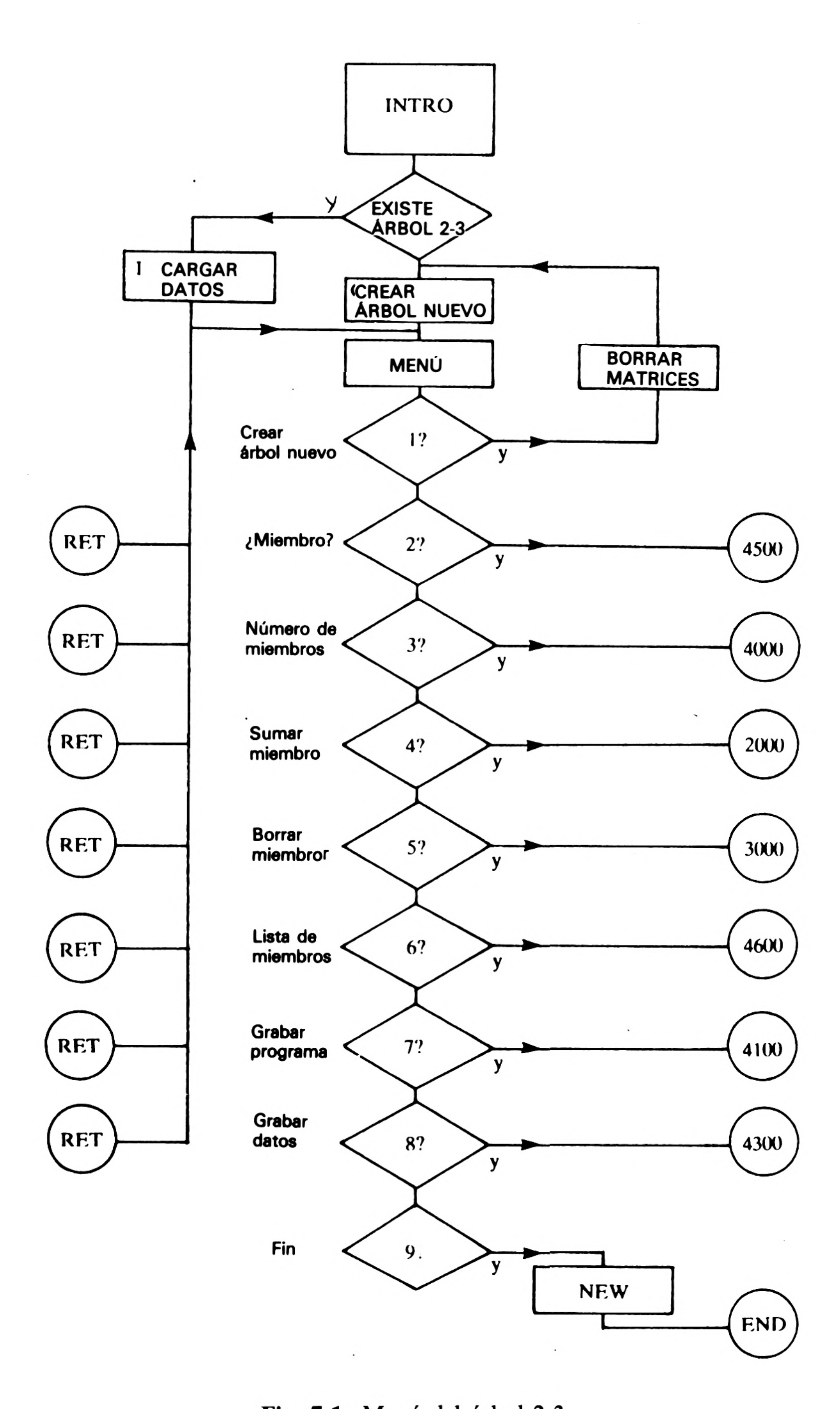
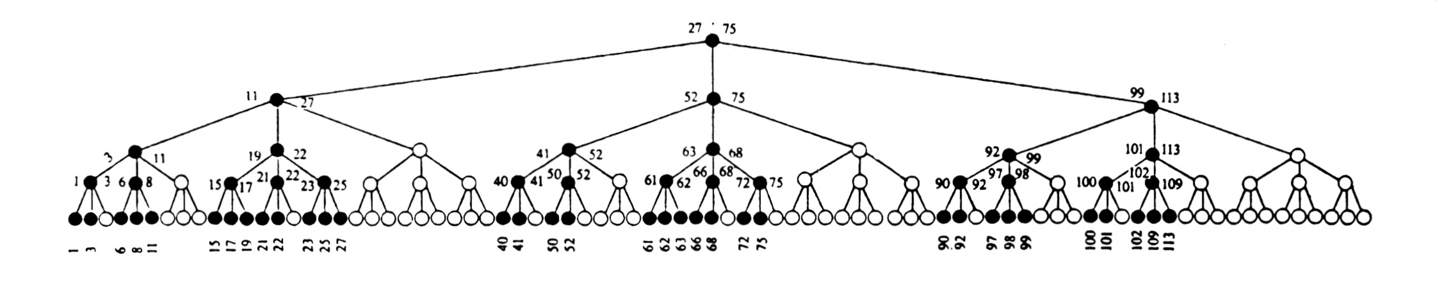


Fig. 7.1. Menú del árbol 2-3.



Observe que el número de la izquierda es la entrada mayor bajo la trayectoria inmediatamente a la izquierda y que el número de la derecha es la mayor entrada bajo la trayectoria inmediatamente a la derecha.

Fig. 7.2. Estructura del árbol 2-3.

Índice

Este índice ha sido confeccionado con respecto a programas o subrutinas particulares que ilustran los diversos usos de sentencias, funciones y cifras, y, por supuesto, no es exhaustivo.

Los números que siguen al título de la subrutina son números de línea, a menos que se indique otra cosa, como, por ejemplo, Intro (Introducción), Ejemplo, SUB, etc. Las referencias a los títulos de las subrutinas y rutinas son de naturaleza general.

¿Tiene que resolver algún problema?

¿Quiere escribir programas?

Este libro único es una recopilación de unas cien subrutinas Amstrad que le ayudarán a resolver sus problemas de programación. Sólo tiene que añadir las subrutinas necesarias a sus propios programas; de esta forma se ahorrará unos días de programación adicional y solucionará muchos problemas que pueden resultarle tediosos.

La cobertura del libro es amplia, e incluye:

- comprobación de los datos introducidos
- clasificación
- almacenamiento y recuperación de datos
- manipulación de imagen de gráficos
- análisis estadístico
- técnicas matemáticas avanzadas, incluida la manipulación de matrices.

En la parte final del libro se incluye una serie de programas completos que ilustra el uso efectivo de estas subrutinas.

Todas las subrutinas están escritas en BASIC para los ordenadores Amstrad 464/664/6128. En los lugares apropiados encontrará comentarios sobre el uso del almacenamiento en cinta o disco.

ISBN 84-86381-14-2

RA — MA
Carretera de Canillas, 144.
28043 Madrid.

AMSTRAD PROGRAMACION PARA SUPERUSUARIOS. W. Johnson ra-ma

AMSTRAD CPC

OCR

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA



https://acpc.me/